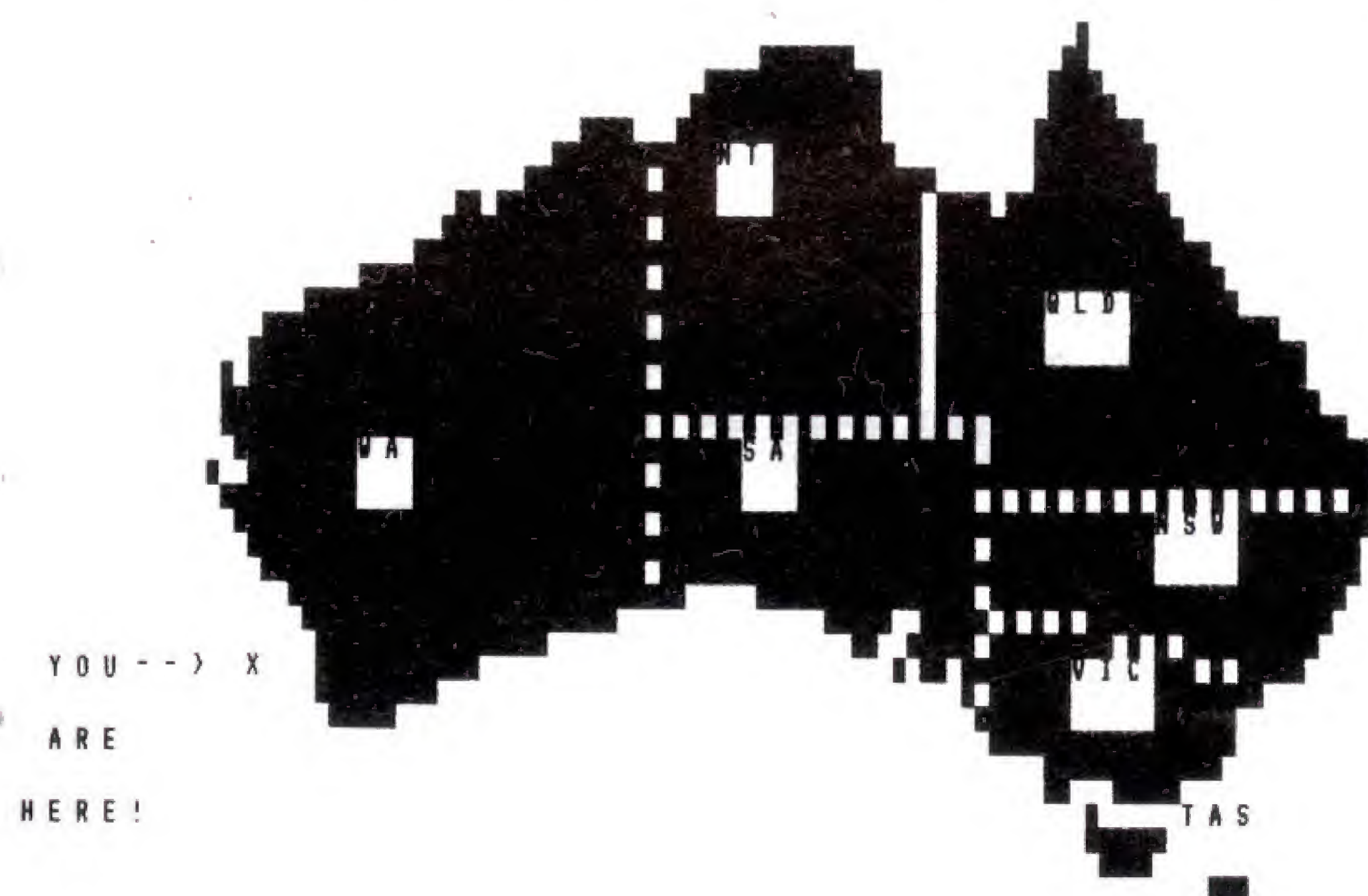


# MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034, AUSTRALIA. TELEPHONE (08) 211 7244    PRICE: AUS. \$3.50; N.Z. \$5.60; U.K. £2.10  
Registered by Australia Post - Publication No. SB02207

Vol. 4, Issue 4, November/December 1983

## AUSTRALIA'S CUP



Also in this issue:

### PROGRAMMING:

Peek, Poke and USR  
Statements Explained

### REVIEWS:

The Adventure System  
The TRS-80 MC-10 Computer

### SOFTWARE:

HI-RES Text—Colour  
Killer Satellite—Colour  
Sirius Adventure—Colour  
Track Racer—Peach

Galactic Battles—Level 1  
Track Racer—Peach  
Sirius Adventure—Peach  
Household Accounting—Peach  
Yahtzee—Level 2  
Bold Type for LP VII—Level 2  
LVAR—Level 2  
GRAFX—Disk  
Space Utility—Model 3  
Source Utility—Model 3  
Household Accounting—  
Model 4  
Space Invaders—MC-10

• TRS-80    • SYSTEM 80    • VIDEO GENIE  
• PMC-80    • HITACHI PEACH  
• TRS-80 COLOUR COMPUTER



# **MOLYMERX**

*Australia's broadest range of software  
for TRS-80's and SYSTEM 80's*

MOLYMERX has the Australian distribution rights for literally hundreds of top grade programs from American, Canadian and British publishers. From games to utilities, from DOS's to Databases, if it's top quality then MOLYMERX almost certainly has it.

Now, MOLYMERX is being distributed in Australia by MICRO-80. To help you chose from the incredibly wide range of programs available, you may purchase a MOLYMERX catalogue. For only \$3.00 you receive over 80 pages of what is virtually an encyclopedia of '80 software plus regular updates for 12 months. The useful information contained in this catalogue is worth many times its cost.

There are now generous BULK BUYING DISCOUNTS of 10% off list price for single orders in excess of \$500 or 15% for single orders in excess of \$1,000. So get together with your friends or User Group members to place a combined order and save yourselves real \$\$\$.

## **EXPANSION INTERFACES FOR SYSTEM 80 and TRS-80 COMPUTERS**

MICRO-80's new family of expansion interfaces for the System 80 and TRS-80 offer unprecedented features and reliability including:

Up to 32K STATIC RAM : to ensure high noise immunity and reliability

Centronics Printer Port: The Systems 80 Expansion Interface has a double-decoded port to respond to both port FD and memory address 37E8H, thus overcoming one of the major incompatibilities with the TRS-80.

RS232 Communications Port: for communicating via modem or direct link to other computers

Single Density Disk Controller: for complete compatability with all Disk Operating Systems

Supports double-sided Disk Drives up to 80 tracks: with a suitable disk operating system such as DOSPLUS, NEWDOS 80 or LDOS, the interface will support single or double sided drives of 35-80 track capacity.

Economical double density: an economical, high quality double-density upgrade will be released shortly to enable you to increase the capacity of your disk drives by 80%.

Real time clock interrupt: provides software clock facility used by most DOS's.

AVAILABLE MARCH 1984, price to be announced shortly. Demand is sure to be heavy. Interfaces will be supplied on a first-come-first-served basis. To avoid disappointment you should write in NOW to be included on the no-obligation waiting list.

## **SYSTEM 80 PRINTER INTERFACE \$99 + \$3.00 p&p**

For those who wish to add a printer to their SYSTEM 80. MICRO-80's new printer interface provides the ideal solution. Double-decoded to both port FD and address 37E8H, this interface overcomes one of the major incompatibilities between the SYSTEM 80 and the TRS-80. Price includes a Centronics printer cable. Operates with Centronics compatible printers including GP-80 and GP-100.

Similar interface is also available for the TRS-80 \$99.00 + \$3.00 p&p

# CONTENTS

## REGULARS

EDITORIAL	2
WHAT YOU HAVE MISSED	4
INPUT/OUTPUT	13

## DEPARTMENTS

KALEIDOSCOPE (COLOUR COMP)	3
PEACH BOWL (HITACHI PEACH)	3
GROUP ONE (MODEL 1/SYSTEM 80)	3
FORM THREE (MODEL 3)	4

## PROGRAMMING

MAKING USE OF PEEK, POKE & USR	5
--------------------------------	---

## REVIEWS

THE ADVENTURE SYSTEM	10
THE TRS-80 MC-10 COMPUTER	12

## SOFTWARE

### COLOUR COMPUTER

HI-RES TEXT <i>Upper/Lower</i>	15 & 36
THE KILLER SATELLITE	15 & 23
SIRIUS ADVENTURE	18 & 24

### PEACH

TRACK RACER	15 & 38
SIRIUS ADVENTURE	18 & 21
HOUSEHOLD ACCOUNTING	16 & 19

### TRS-80/SYSTEM 80

AUSTRALIA'S CUP (L2/16K)	13 & 39
YAHTZEE (L2/16K)	18 & 28
BOLD TYPE FOR LP VII (L2/12K)	18 & 27
LVAR (L2/32K)	15 & 44
GRAFX (DISK 32K)	14 & 43
SPACE UTILITY (DISK/48K)	18 & 27

### OTHER

SOURCE UTILITY (MOD3/DISK)	14 & 46
HOUSEHOLD ACCOUNTING (MODEL 4)	16 & 31
SPACE INVADERS (TRS-80 MC-10)	18 & 26
NEXT MONTH'S ISSUE	47
CASSETTE/DISK EDITION INDEX	48
ORDER FORM	48

## ABOUT MICRO-80

EDITOR: IAN VAGG

MICRO-80 is an international magazine devoted to the Tandy TRS-80 Model 1, Model III and Colour microcomputers, the Dick Smith System 80/Video Genie and the Hitachi Peach. It is available at the following prices:

	12 Months	Single Copy
MAGAZINE ONLY	\$ 36.00	\$ 3.50
CASSETTE SUBSCRIPTION	\$ 96.00	\$ 6.00
DISK SUBSCRIPTION	\$125.00	\$10.00 (disk)

MICRO-80 is available in the United Kingdom from:

U.K. SUBSCRIPTION DEPT. 24 Woodhill Park, Pembury, Turnbridge Wells, KENT TN2 4NW		
MAGAZINE ONLY	£16.00	£1.50
CASSETTE SUBSCRIPTION	£43.00	£N/A
DISK SUBSCRIPTION	£75.00	£N/A

MICRO-80 is available in New Zealand from:

MICRO PROCESSOR SERVICES, 940A Columbo Street, CHRISTCHURCH 1 NZ. Ph. 62894		
MAGAZINE ONLY	NZ\$ 59.00	NZ\$ 5.60
CASSETTE SUBSCRIPTION	NZ\$130.00	NZ\$ 7.50
DISK SUBSCRIPTION	NZ\$175.00	NZ\$15.00

MICRO-80 is despatched from Australia by airmail to other countries at the following rates:

	(12 MONTH SUB) Magazine	Cass Sub	Disk Sub
PAPUA NEW GUINEA	Aus\$53.50	Aus\$115.50	Aus\$148.50
HONG KONG/SINGAPORE	Aus\$58.00	Aus\$122.00	Aus\$157.50
INDIA/JAPAN	Aus\$64.00	Aus\$129.00	Aus\$165.00
USA/MIDDLE EAST/CANADA	Aus\$73.00	Aus\$140.00	Aus\$177.00

Special bulk purchase rates are also available to computer shops etc. Please use the form in this issue to order your copy or subscription.

The purpose of MICRO-80 is to publish software and other information to help you get the most from your TRS-80, System 80/Video Genie or Peach and its peripherals. MICRO-80 is in no way connected with any of the Tandy, Dick Smith or Hitachi organisations.

**WE WILL PAY YOU TO PUBLISH YOUR PROGRAMS:** Most of the information we publish is provided by our readers, to whom we pay royalties. An application form containing full details of how you can use your microcomputer to earn some extra income is included in every issue.

**CONTENT:** Each month we publish at least one applications program in BASIC for each of the microcomputers we support. We also publish Utility programs in BASIC and Machine Language. We publish articles on hardware modifications, constructional articles for useful peripherals, articles on programming techniques both in Assembly Language and BASIC, new product reviews for both hardware and software and we printer letters to the Editor.

**COPYRIGHT:** All the material published in this magazine is under copyright. This means that you must not copy it, except for your own use. This applies to photocopying the magazine itself or making copies of programs on tape or disk.

**LIABILITY:** The programs and other articles in MICRO-80 are published in good faith and we do our utmost to ensure that they function as described. However, no liability can be accepted for the failure of any program or other article to function satisfactorily or for any consequential damages arising from their use for any purpose whatsoever.

MICRO-80 is Registered by Australia Post — Publication No. SBQ2207

**AUSTRALIAN OFFICE AND EDITOR:** MICRO-80, P.O. Box 213, Goodwood, S.A. 5034. Tel. (08) 211 7244

**U.K. SUBSCRIPTION DEPARTMENT:** 24 Woodhill Park, Pembury, Turnbridge Wells, Kent TN2 4NW

**TYPESETTING & MAKE-UP:** Formgraphic, 117 Wright Street, Adelaide, S.A. 5000. Tel. (08) 211 7866

**PRINTED BY:** Shovel & Bull Printers, 379 South Road, Mile End, S.A. 5031

**PUBLISHED IN AUSTRALIA BY:** MICRO-80, 433 Morphett Street, Adelaide, S.A. 5000

# EDITORIAL

Welcome to our new-look MICRO-80 magazine. We hope you find the new layout more readable than the old. For those interested in the mechanics of magazine production, our type setting is now being done on a Compugraphic 80 Electronic phototypesetting machine. This is effectively a multi-user computer/word processor which produces its output photographically. The equipment is pretty much state-of-the-art and is another product of the microprocessor revolution. Just in case you think we must have won the lottery, the type setting equipment is not ours but is the property of Formgraphic, a very progressive type setting house located nearby.

Quite apart from making the magazine easier to read, the new type setting fits double the amount of information on each page thus enabling us to reduce the paper used by almost 25% (the Listings still occupy the same space as before). So when future issues look a little slimmer, don't fret, you will still be receiving the same amount of information. The reduction in paper content is aimed at reducing the magazine's production costs. It is now 2 years since the price of MICRO-80 was last set. In that time, production costs have increased dramatically. Even with the reduction in material costs we have now effected, we must reluctantly increase the selling price of MICRO-80. The new rates which become effective immediately, are published on the index page. The price increase is modest and will enable us to continue to improve our magazine and adapt it to meet the changing requirements of our readers. Of course, you will not be required to pay the new rates until your subscription comes up for renewal. Incidentally, we do send Reminder Notices when your subscription nears its expiry date.

There have been several changes in key staff at MICRO-80. Ryzard Wiwatowski, the Editor for the past 12 months has resigned as has Charlie Bartlett, the Software Editor for the past 2½ years. Charlie is moving to sunny Queensland. I am sure we all wish both men well in their future careers. Ian Vagg has assumed the role of Editor once more whilst Ed Grigonis who operates a Model 1 system with disk drives and is an active member of the Adelaide Micro Users Group, has become Software Editor. The departure of Ryzard and Charlie caused no small disruption in our production schedule. The result is the issue you are reading now. It is a double-sized issue (after allowing for type setting) and encompasses both November and December 1983 editions. It is also rather late. The delayed production ran us slap-bang into the Christmas season and the subsequent annual close down of our printers. It will take a month or two until we are once more back on schedule so please accept our apologies for any inconvenience caused.

Much has been happening of late in the '80 field. Tandy has announced several new products including a portable Model 4 known as a Model 4P. This is essentially a Model 4 in a more compact package with 9" monitor, half-height disk drives and separate keyboard which all packs away into a large carrying case. The complete unit weighs in at about 12kg and

with 2 disk drives and 64K of RAM sells in the U.S.A. for \$US1,799.00. Since the 4P has not yet been announced in Australia its price here is unknown but will probably be \$2,600-\$2,700, now that the Model 4 itself has been reduced to \$3,000. There seems little doubt that like radios and calculators before them, computers are going to become portable. The main impediment to this process is the bulk and weight of a readable sized display. Once flat screen or large liquid crystal displays become available at a suitable price the desk-top microcomputer as we know it today is certain to become a quaint relic of the past. The only question now seems to be, how quickly will it all happen?

On a less buoyant note, we have heard from a usually reliable source that EACA, the Hong Kong firm that manufactured System 80/Video Genie, has ceased trading altogether. This will mainly affect those System 80/Video Genie owners who are planning to upgrade their systems to include disk drives or a printer, since printer interfaces and expansion interfaces are no longer available. Fortunately, here at MICRO-80 we had already commenced design work on a new expansion interface and also a printer interface, when the news broke. The interfaces will be available for sale towards the end of February 1984. The expansion interface (there is also a TRS-80 version) includes a printer port, RS232 port, floppy disk controller and up to 32K of static RAM. The use of static RAM gives high noise immunity and removes many sensitive timing problems associated with the dynamic RAM used in the early designs. The printer interface is decoded to both port FD (standard System 80/Video Genie printer port) and also to memory address 37E8H, the address used in the TRS-80 Model 1. In this way Tandy software which drives the printer port directly will operate satisfactorily on the System 80. The expansion interface has the same arrangements for its printer port, too.

The demise of EACA is somewhat symptomatic of the changes taking place in the '80 world. The proliferation of different brands of microcomputer, each more dazzling and spectacular than the last, has reduced the total amount of support available for any one brand. In particular, older machines such as the TRS-80/System 80 are suffering badly. There are many fewer organisations catering for these machines and even their original distributors Tandy and Dick Smith have markedly reduced their levels of support in favour of later models or different machines altogether. At the same time, these computers still have a good deal of useful life left in them and there are literally tens of thousands of owners who have a large personal investment in them through acquired knowledge and programs, both written and purchased. We believe that the TRS-80/System 80 computers are far from finished and it is time that owners drew together to assist each other even more as commercial support wanes. MICRO-80 intends to become an even more important focal point for this support. The development of new expansion interfaces for the TRS-80 and the System 80 and the distribution of Molymerx software in Australia are just two of the ways in which we have increased our support for the owners of '80 computers. MICRO-80 itself will also change to better reflect the interests of its readers. We will for the first time encourage the placement of advertisements for relevant products in our magazine. Rates will be much lower than in the general computing magazines

so that small, specialist suppliers will be able to afford to advertise. We will shortly launch a New Products column containing information about relevant new products for '80 computers. In these ways, MICRO-80 will become a more complete reference to all things '80 and help maximise the diminishing support. While some of this may not be of great relevance to overseas readers we are sure that the changes in content which are planned will be very appealing.

Over the years we have received many comments from readers, both positive and negative. Analysis of these comments indicates that whilst the newer computer owners find the material in MICRO-80 to be pretty much what they want, the more experienced owners find fewer articles and programs to suit their tastes. One of the most consistent complaints is that we publish too many games (usually expressed by the disgruntled as, "All you publish is games"). There are two interesting observations to be made here. Firstly, we do publish many other types of programs besides games. Secondly, the programs we publish represent a fair cross-section of those being written (and presumably used) by Readers so we are probably catering to the "average" taste. That said, we accept that we do not publish many "Serious" programs in terms of relatively complex applications such as Accounting Systems, Data Management etc. Certainly, those of our critics who suggest we should publish full-featured working accounting systems for example, are being somewhat unreasonable. Such software sells for \$2,000 or more. It seems a little much to expect to be given it for the price of a magazine subscription. The reason these programs cost so much is the very considerable amount of work, often amounting to man-years, required to write, test, debug and maintain them. Speaking personally, if it were my business at stake, I would make sure that I paid the full rate for a properly supported program because, when things go wrong I want them fixed quickly. Nevertheless, there is considerable scope to increase our coverage in the serious applications area. We will therefore, introduce a number of changes over the next few issues. Firstly, we will support two data base programs, the Tandy Profile series and ENbase from Southern Software. The first because they are useful programs that are very widely used, the second because we believe ENbase to be the most powerful data base available for the '80 computers. Our support will take the form of articles to assist you to understand the concepts behind the programs and how to get the best form them and also "Listings" of specific applications which you can use just as you would a BASIC program. We would welcome (and pay for!) submissions by our readers.

In a similar vein, we will support VISICALC and SUPER VISICALC with articles, templates etc.

We will publish more in-depth reviews of serious applications software such as word processors, graph generators, data bases etc. Games playing can also be a serious business for the real enthusiast so Ed Grigonis has undertaken to write a series of detailed reviews on many of the better quality games. Ed's first contribution on the Adventure System is in this issue from which you will see that he writes very high standard reviews indeed.

One of the more neglected fields in computer magazines of all kinds, is that of utilising the graphics capabilities of dot-matrix printers. The ubiquitous Epson

MX-80 has spawned a host of cheap compatibles in its wake. There seems to have been few articles published anywhere to assist the owner unravel the jinglish in the instruction manuals, let alone make use of the extremely versatile bit-graphics capabilities of these printers. MICRO-80 will attempt to rectify these omissions by publishing articles, programs etc which will

assist the very many owners of such printers to improve their usefulness.

Even with the increased emphasis on "Serious" computing planned, we will not neglect our less experienced readers and will continue to provide articles and programs to suit the new and intermediate owner.

We hope you will find reading our "New-Look" MICRO-80 as exciting and interesting as we find the prospect of producing it. We believe that, whatever your background and experience, each issue will contain something to interest and challenge you.

## DEPARTMENTS KALEIDOSCOPE

A major drawback in using the high resolution screen on the Colour Computer is the difficulty encountered when you try to mix the text with the graphics. One way of solving this problem was presented in last month's issue, but the WRITER program in this issue by Geoffrey Williamson demonstrates a much more flexible and useful solution to the problem. Those of you with Tandy's Editor/Assembler Plus who wish to type in the source code from the magazine will notice that a 'few lines' have been left out between lines 1500 and 12,040. This omission is quite deliberate and in the interest of saving space — the source lines omitted are simply the multitude of FCB's that define the bytes of the character table and these can be just as easily read from the Hex Dump. For reference, source line 1410 corresponds to the Hex Dump at \$30C0. For those of you entering the Hex Dump, the memory addresses commencing at \$3000 are only a suggestion since you cannot place the code starting at zero in reserved RAM. However, the program is written in position independent code and you can locate the program anywhere you find it convenient.

### WATCH BASIC AT WORK

Normally, the text video display memory on the Colour Computer is located at \$400-\$5FF. The actual values stored here determine which characters are displayed on the screen of your TV set. However, the combination of SAM and VDG potentially allow you to set the location of display memory to anywhere in memory aligned on a 512 byte boundary. For those of you who are interested in how a BASIC program or the interpreter itself works, this capacity provides a rather unique opportunity for a visual demonstration.

In the September '83 issue, the piece on conserving memory mentioned the frivolous and extravagant use of the CLEARED string space by the BASIC interpreter. The following program provides a graphic illustration of how the string

space is managed by the BASIC interpreter.

The program, first of all, reserves some high memory for a machine language subroutine and POKES it into memory. The subroutine modifies the display offset register in the SAM chip and allow you to display any portion of the 16K RAM. The routine is called in line 40 and sets the display memory to addresses \$300 — \$3FF (in fact, any parameter in this range would set the display to this page of memory). Lines 50 to 70 do some string manipulation which you can watch on your screen (the subroutine at line 90 is used to slow the process down to a pace with which the eye and brain can cope).

You should see that the interpreter uses up free string space at a rather rapid rate until it runs out of space. When this happens a garbage collection routine compacts all the currently active strings at the top of string space and then releases the remainder and returns to running the program. With a very large amount of string space, this garbage collection routine can sometimes take several minutes to executive and if the actual amount of free string space is small, then it will take place frequently, slowing down the speed at which the BASIC program is running.

The machine language subroutine is completely relocatable but does use one ROM call to retrieve the parameter passed by the USR function. This address (\$B3ED) may need to be changed in future revisions of the Colour BASIC ROMs. You can use this technique to look at the operation of other BASIC statements by changing the program lines 40-80 and you will probably need to look at different areas of memory to see where the action is taking place. The only limitation so far seems to be that when the interpreter uses the normal text page, it sets the display offset register in the SAM to show the normal text page. However, with a little trial and error, you can use this method to explore the workings of your Colour Computer and its BASIC interpreter.

```
10 CLEAR 200,&H3F00
15 DIM B$(26)
20 A=&H3F00 : DEFUSR0 = A
30 FOR I=0 TO 26 : READ A : POKE A+I,D : NEXT I
40 A=USR0(&H3E00) : A$=""
50 FOR I=0 TO 26
60 A$=A$+CHR$(I+63) : B$(I)=CHR$(32) : GOSUB 90
70 NEXT I
80 A=USR0(&H400) : END
90 FOR J=0 TO 750 : NEXT J : RETURN
100 DATA 77,38,23,189,179,237,70,198
110 DATA 7,142,255,198,70,36,6,48
120 DATA 1,167,128,32,2,167,129,90
130 DATA 38,242,57
```

## PEACH BOWL

If you own a Peach and are a new subscriber or have renewed your subscription with or subsequent to the first issue of Volume 4, then you are entitled to the new free software offer. However, the SOFTPAK program library is not suitable for the Hitachi Peach — instead, we offer our Peach readers the choice of one of three commercial games, viz. Peach Invaders, Ghost Gobbler or Scrambler. We are now at the stage where we wish to prepare and distribute this software gift but have encountered two difficulties:

1. Our mailing list does not identify all of our Peach subscribers.
2. We have no record of which particular game you wish to receive.

If you are a Peach owner already entitled to receive the new free software or who will become eligible by renewing your subscription at some stage during Volume 4, then please drop us a line as soon as possible with the following information:

- (i) Your name and address
- (ii) Your subscription expiry issue
- (iii) Your selection of one of the following:  
A—Peach Invaders B—Ghost Gobbler  
C—Scrambler

Please direct this information to:  
PEACH FREE SOFTWARE,  
MICRO-80,  
P.O. BOX 213,  
GOODWOOD, S.A. 5034

## GROUP ONE

Many of our readers with disk systems will appreciate the GRAFX utility in this month's issue. A number of people have recently pointed out that many Level 2 BASIC programs utilizing machine language subroutines more often than not do not work with Disk BASIC (see Input/Output and below). The reasons for this are fairly obvious. Most Z80 machine code contains local absolute address references and cannot be easily relocated. Secondly, on 16K Level 2 machines, these subroutines are usually placed at the top of memory (i.e. below address 8000 Hex) which usually conflicts with the program storage area in Disk BASIC in the case of all but the smallest of BASIC programs. The program RECALL on the SOFTPAK disk offers one type of solution to this problem. Another approach is the following from one of our readers, Mr. Wilson of Toronto:

### MODIFICATION OF 'GOLF' FOR DISK USAGE

I recently added a disk drive to my System 80 and then, of course, set about transferring all my tape based programs to disk. With most BASIC programs this

presents no problem at all. However, any such program which contains a machine language subroutine will require some modifications. One program of this type which I have converted is GOLF which was published in the July 1983 issue of MICRO-80. The changes I made may be of interest to some other readers. This is particularly true because, although articles are often published which point out that changes are needed and some describe the disk instruction DEFUSR, I have not seen any which explain in any detail all the changes needed to convert a given program.

The DEFUSR function was described in the August 1983 issue of MICRO-80. However, a brief recap of its use is given here again for completeness.

In tape-based Level 2 BASIC the entry address of a machine language subroutine is divided into its least and most significant bytes which are POKEd into locations 16426 and 16527 before the routine is called using the A=USR(O) call.

In disk BASIC the entry address is defined by DEFUSRn = address where address is decimal or hexadecimal. The number n in the DEFUSRn statement can be 0 to 9 thus allowing up to 10 machine language subroutines to be defined. The calls to the subroutines are made by A=USRn(x). More details on DEFUSR and USR will be found in your DOS manual.

In the GOLF program there is a short machine language subroutine defined in statements 10 to 60. This routine actually stores the screen image of the GOLF hole so that it can be restored to the screen later in the program. The machine language statements are stored in the dummy string LL\$. The address of LL\$ is then POKEd in line 60.

The machine language routine works by storing the 1024 screen data values from addresses 15360 to 16383 into memory locations 30720 onwards. In a 16K machine the locations are above the addresses needed for the BASIC program so they are in a safe location. However, when disk BASIC such as DOSPLUS 3.4 is used, space up to about 21K is used by disk BASIC and GOLF then stores up to about 35K. Thus the machine language routine uses locations in which BASIC program statements are stored. This will cause chaos to say the least! The solution is to change the routine so that it uses much higher locations in memory.

The address which must be changed in the machine language routine is given by the sequence 0,120 which appears in line 10 and line 20 DATA statements. In line 10 it is items 14 and 15 and in line 20 it is items 7 and 8. 0,120 defines a hex location 7800 (or decimal 30720). My machine has 48K RAM and thus addresses up to 65535. I therefore decided to place both the machine language routine itself and the storage locations used by the routine above 63000. I used location FA00 (or decimal 64000) in the routine as a storage location. The value FA00 translates into data values 0,250. Thus, to effect the change, alter the value 120 in item 15 of line 10 and item 8 of line 20 to 250.

The routine itself I decided to locate at 63500 (F80C in hex). This is achieved by replacing lines 30 to 60 of the original program by:

```
30 FOR I=63500 TO 63529
35 READ LO
40 POKE I-65536,LO
45 NEXT I
50 DEFUSR=63500
```

The final point to note is the POKE address in line 40. Because the System

80 can only handle integers up to 32767 it is necessary to use negative integers for the addresses above this value in POKE and PEEK statements.

With these two changes GOLF Now runs as it used to on tape.

## WHAT YOU HAVE MISSED

Set out below is a list of some of the programs published in early issues of MICRO-80 magazine. Back issues are available for \$2.50 each or at the annual subscription rate for 12 or more copies. Cassette editions are available for all issues for \$4.00 each whilst DISKS are available for all issues FROM SEPTEMBER 1981 onwards. For 12 or more magazines with cassette/disks ordered at the same time, the relevant annual subscription rate applies. Programs for the Hitachi Peak/TRS-80 Colour Computer were first published in the April 1982 issue. Complete indices to the first three volumes of MICRO-80 magazine are included in the December 1980, December 1981 and the August 1983 edition.

### ISSUE 10—SEPTEMBER 1980\*

ESCAPEE	(L1)
THE WORLD	(L1)
CUP '80	(L1)
CUP '80	(L2)
TRIANGLE	(L2)
THE WORLD	(L2)
SOLVER	(L2)
LOTTO PREDICTOR	(DB)

### ISSUE 20—JULY 1982

SHARE GRAPH	(L1)
CHEQUE BOOK DATA FILE	(L1)
BLOWFLY	(L2)
MILEAGE CALCULATOR	(L2)
CONVERSIONS	(L2)
STAR SHOOT	(L2)
BINGO	(L2)
GENIUS	(L2)
DISK INDEX	(DISK)

### VOLUME 3 NO. 7—JUNE 1982

UNIT CONVERSIONS	(CC/PEACH)
NORMAL DISTRIBUTION	(CC/PEACH)
MICRO GRAND PRIX	(L2)
PASSWORD	(L2)
PASSWORD CHANGE PROGRAM	(L2)
OTHELLO	(L2)
LOAN CALCULATION PACKAGE	(L2)

L1—Level 1

L2—Level 2

CC—Colour Computer

HP—Hitachi Peach

\*Issue incorrectly labelled August.

The following back issues of MICRO-80 magazine are still available:

	'79	'80	'81	'82	'83
Jan	—	✓	✓	✓	—
Feb	—	✓	X	✓	—
Mar	—	✓	✓	✓	—
Apr	—	X	✓	✓	—
May	—	✓	✓	✓	—
Jun	—	X	✓	✓	—
Jul	—	✓	✓	✓	✓
Aug	—	X	✓	✓	✓
Sep	—	✓	✓	✓	✓
Oct	—	X	✓	X	✓
Nov	—	X	✓	—	—
Dec	✓	✓	✓	—	—

— means never published

✓ means issue available

X means issue out of print

## FORM THREE

NEWDOS 80 provides a copy of the original Radio Shack Editor/Assembler modified so that source files can be saved to or loaded from disk. However, on the Model 3, Apparat chose to drop support for cassette tape so that you cannot load source files from cassette at all. This can be quite frustrating if you wish to modify a source file you may only have on tape. The Source utility in this month's issue will overcome this problem and save you having to tape the source code in a second time.

### BMON ON THE MODEL 3

A number of readers have enquired about using BMON on the Model 3, without much success. Interestingly, the Adelaide Micro User Group published in their October newsletter some patches developed by one Tony Domigan to make BMON work on the Model 3. His item is reproduced here with permission for the benefit of our readers.

Eddy Paay's BMON will not work on the Model 3 because it uses the Model 1's keyboard caller address (03E3H), and jumps to BASIC via 06CCH. Rather than just patch the old caller with 3024H I have reworked some of Eddy's code to patch the current keyboard caller thus allowing BMON to work in Newdos, Ldos and TRSDos Disk BASICs as well as Model 3 BASIC. Furthermore, all cassette routines will prompt you for the baud rate to use and the ASCII character will now be displayed alongside the hex character in the edit mode.

Edit characters (below) enclosed in brackets, e.g. (FX) are for the 48K version only. If you are using a 32K BMON then substitute 'BX' and for the 16K version use '7X'.

- (a) Reserve Memory  
(b) Load BMON thru SYSTEM ... BMON  
(c) In place of answering '/ENTER' substitute ... /64464 (48K), /48090 (32K), /31696 (16K)
- Edit address FB99/BB99/7B99 and enter ... CD, C9, 01, 21, 25, (FB), CD, 1B, 02, 2A, 16, 40, 22, C7, (FB), 21, C6, (FB), 22, 16, 40, 01, 18, 1A, C3, AE, 19, CD, C9, 01, CD, 42, 30, C9, CD, 33, 00, E5, 7E, 2A, 20, 40, C3, OC, FC, CD, 24, 30, B7, C8, FE, 02
- Edit FB4E/BB4E/7B4E and enter 43, 54, 52, 4C, 3E, 20, 20, 42
- Edit FB60/BB60/7B60 and enter 29, OD
- Edit address FBFB/BBFB/7BFB and enter ... 21, E8, (FB), CD, 1B, 02, 21, 00, 50, CD, 60, 00, CD, 42, 30, 18, OC, 77, 23, 3E, 20, 77, 23, 22, 30, 40, E1, C9, 00
- Edit address FD6E/BD6E/7D6E and enter CD, A8, (FB)
- Edit address F9C1/B9C1/79C1 and enter CD, B4, (FB)
- Select (B)asic and execute the BASIC line applicable to your BMON version.  
10 POKE - 2558, 187: POKE - 2557, 251' (48K)  
10 POKE - 18942, 187: POKE - 18941, 187' (32K)  
10 POKE 30209, 187: POKE 30210, 123' (16K)



9. Cassette users enter CTRL-B (shift/down arrow/B) to enter BMON and create a system tape of the modified BMON.

BMONStart	End	Entry	Addresses
16K	7210	7EFE	7B99
32K	B210	BEFE	BB99
48K	F210	FEFE	FB99

Disk users should re-boot DOS and transfer the program using the 'DUMP' command.

—Adelaide Micro User News, October, 1983.

#### DEFUSR PROBLEM

Some programs intended for Level 2 systems will cause problems when you try to run them on the Model 3. For example, Andre Marino reports the following difficulty:

"I am writing about a program you have already published two months ago (August, 1983). The program is DEFUSR. I am having problems in getting the program to load. The program loads for a brief second and then the screen scrolls up with a continuous flow of question marks. I have a TRS-80

Model 3 48K cassette based computer. My belief, through some experimentation, is that the program is in a bad area of memory, but have found no way to make the program work. I would appreciate it if you could help me out with this problem."

The memory from 4040H to 404FH is not a good place to put machine language programs on the Model 3. Although the Level 2 scratch pad areas used by the Model 1 and Model 3 are the same, most of the reserved RAM area used by the Disk Operating System is quite different. Parts of the Model 1 DOS reserved RAM is used to implement other features in the basic Level 2 mode of the Model 3 and more low memory is reserved for use by the DOS. This means that those machine language programs residing in Model 1 DOS reserved RAM locations will probably not work on a Level 2 Model 3. There are two solutions to this problem. The first involves protecting some high memory and moving the program to high memory (suggested in the Form

Three column of the same issue). The second is to 'hide' the program between the reserved RAM and the start of the BASIC program storage area by moving the latter to a higher memory location.

If the particular machine language program is not relocatable and contains local absolute address references, then these must all be changed to reflect the program's new location in memory. If the source code is available then this is best done by reassembling the program at the new memory location by changing the ORG statement. If only the object machine code is available, then this can be a long and complex task which must be done by hand. Fortunately, the DEFUSR program does not require any such changes as it is relocatable. The first method has the additional disadvantage that you must protect high memory each time before using the program.

The second method is more elegant in the case of **relocatable** machine language programs and can be achieved by the following code:

```

00100 ; DEFUSR for the Model 3
4F00      00110      ORG      4F00H      ; Entry /20224
4F00 2AA440      00120      LD      HL,(40A4H)      ; Start of BASIC pointer
4F03 225C41      00130      LD      (415CH),HL      ; Disk BASIC exit for DEFUSR
4F06 11194F      00140      LD      DE,DEFPRC      ; The DEFUSR code address
4F09 1A      00150      LOOP      LD      A,(DE)      ; Move the code to where
4F0A 77      00160      LD      (HL),A      ; BASIC programs normally
4F0B 13      00170      INC      DE      ; start
4F0C 23      00180      INC      HL
4F0D B7      00190      OR      A
4F0E 20F9      00200      JR      NZ,LOOP      ; Loop until finished
4F10 22A440      00210      LD      (40A4H),HL      ; Set new Start of BASIC
4F13 CD4D1B      00220      CALL      1B4DH      ; Do a 'NEW' to setup the
                                00230      ; remaining BASIC pointers
4F16 C3191A      00240      JP      1A19H      ; Return to BASIC
4F19 CF      00250      DEFPRC      RST      8      ; DEFUSR code
4F1A C1      00260      DEFB      0C1H
4F1B CF      00270      RST      8
4F1C D5      00280      DEFB      0D5H
4F1D CD3723      00290      CALL      2337H
4F20 E5      00300      PUSH      HL
4F21 CD7F0A      00310      CALL      0A7FH
4F24 228E40      00320      LD      (408EH),HL
4F27 E1      00330      POP      HL
4F28 C9      00340      RET
4F29 00      00350      NOP
4F00      00360      END      START      ; Mandatory terminator

```

This program will tuck DEFUSR between reserved RAM and the BASIC program storage area. This technique could also be used to place other programs here with some precautions. The program should be relocatable and must not contain a zero within its code (since the particular loop that moves the program code terminates when a zero is encountered — a different loop structure could be used). If you don't have an assembler then the following BASIC program will load the program into memory and run it:

```

10 'DEFUSR for the Model 3
20 POKE 16526,0 : POKE 16527,79 ' Set USR entry point
30 A=20224 'Start Address
40 FOR I=0 TO 41 : READ D :POKE A+I,D : NEXT I
50 X=USR(0)
60 DATA 42,164,64,34,92,65,17,25,79,26,119,19,35,183,32,249
70 DATA 34,164,64,205,77,27,195,25,26,207,193,207,213,205,55,35
80 DATA 229,205,127,10,34,142,64,225,201,0

```

Be warned, this will destroy any resident BASIC Program so CSAVE the program **before** you RUN it.

## PROGRAMMING

### AN EXPLANATION OF HOW TO MAKE FULL USE OF THE PEEK, POKE AND USR STATEMENTS

by Gordon S. Thomas

The object of this article is to explain the use of the PEEK, POKE and USR statements and functions in Level 2/Model 3 BASIC following a request in the "Readers' Requests" section in Micro-80 Vol. 3 No. 10 (September 1982). The article assumes no previous knowledge of the uses of these statements.

Unless otherwise stated, all information contained herewith is equally applicable to TRS-80 Models 1 and 3

cassette and disk systems (and all other software compatible computers) with any memory size. It is not applicable to Level 1 BASIC computers since they do not have these statements.

The computer's memory is made up of two main types of memory, which are Read Only Memory (ROM) and Random Access Memory (RAM). ROM is where the Level 2/Model 3 BASIC interpreter is stored. ROM cannot be changed by any software (i.e. written to).

RAM, on the other hand, can be changed and it is here that all user programs and data are stored (until the power is disconnected). Therefore, to modify the computer's operation in any manner by software requires that the contents of RAM be changed.

The Z80 microprocessor contained in the computer is capable of interacting with 65536 memory locations. Depending on the computer, anywhere between 12K and 14K of these memory locations are used by the ROM with the rest being used by the RAM. Each one of these memory locations is assigned a number, called an "address". It is these addresses which the PEEK function and the POKE statement require in their respective syntax.

There are various, different ways of referring to these addresses. They can be numbered using the decimal system (which is just our normal everyday counting system) or they can be numbered using the hexadecimal system, which is what the computer uses. (This is not strictly correct but is adequate for the purposes of this article).

The hexadecimal system uses the digits 0-9 and the letters A-F to designate the decimal numbers 0-15 respectively. Instead of each place in a number being a power of 10 (as it is in the decimal system) the hexadecimal system has every place representing a quantity of a power of 16, e.g. 38 (decimal) =  $(3 \times 10^1) + (8 \times 10^0)$   
 30 (hex) =  $(3 \times 16^1) + (8 \times 16^0)$   
 = 56 (dec)  
 38 (dec) =  $(2 \times 16^1) + (6 \times 16^0)$   
 = 26 (hex)

**PEEK:** If we want to know what value is stored in a particular address all we have to do is type: PRINT PEEK (address) e.g. to find out what is stored in the top left hand corner of the screen (address 15360) we would type  
 PRINT PEEK (15360)

This will return a decimal number which represents what is contained in address 15360. Therefore if the top left hand corner of the screen contains the letter "B" then the decimal value 66, which is the ASCII code for the letter "B", will be returned.

Any address in the computer's memory can be PEEKed (i.e. from 0 to top of RAM). The top of RAM addresses for the various memory sizes are as follows:

Mem Size	Hex	Dec
16K	7FFF	32767
32K	BFFF	49151
48K	FFFF	65535

In order to PEEK any address above 32767 it is necessary to subtract 65536 from the address in question. e.g. to display the contents of 40000, type  
 PRINT PEEK (-25536)  
 since  $40000 - 65536 = -25536$ . The hexadecimal equivalent of 40000 is 9C40H since  $9C40H = (9 \times 16^3) + (12 \times 16^2) + (4 \times 16^1) + (0 \times 16^0) = 40000$  where H indicates that it is hexadecimal quantity.

However, if you type (for Disk BASIC only)  
 PRINT &H9C40 (syntax for 9C40 hex) the computer will respond with -25536, i.e. the computer takes care of the conversion process. This is why many programmers prefer to use hexadecimal when referring to memory addresses — it requires no additional calculations to determine what number to use to designate a particular address. In general, to PEEK any address in the computer's memory, type  
 PRINT PEEK (X + 65536 \* (X > 32767))

For  $X > 32767$  the expression  $(X > 32767)$  will be TRUE, resulting in -1.

This effectively subtracts 65536 from the address.

For  $X < 32767$  the expression  $(X > 32767)$  will be FALSE, resulting in 0.

This will not affect the address in any way.

The same rules apply for the addresses in the POKE statement.

**POKE:** If we want to change the contents of a particular memory location we use the POKE statement. Its syntax is POKE address, value e.g. to store a "1" in the top left hand corner of the screen we would type  
 POKE 15360, 49  
 (49 is the ASCII code for the number "1")

The POKE statement is useful for loading small machine language routines into memory to be accessed from BASIC. It also has many other uses. Some of the most common are:

1. setting memory size from BASIC (addresses 16561 - 16562)
2. loading graphics character onto the screen (addresses 15360 - 16383)
3. disabling the (BREAK) key in Cassette BASIC (addresses 16396 - 16397)
4. pointing to a USR routine in Cassette BASIC (addresses 16526 - 16527)

In order to be able to fully utilize the capabilities of the POKE statement it is necessary to have a basic understanding of the terms "least significant byte" and "most significant byte". When we see the number 327 in everyday life we all know what it represents, since we are used to dealing with decimal quantities. The 3 can be thought of as the most significant digit and the 7 can be thought of as the least significant digit. In hexadecimal we group two digits together and call it a byte.

e.g. for 9C40H (40000 decimal) the most significant byte is 9CH with the least significant byte being 40H. The decimal equivalents of 9CH and 40H are 156 and 64 respectively. There is an alternative way of determining these numbers. Note that 156 is actually stating how many whole lots of 256 (decimal) that there are in 40000, and that 64 is stating how many lots of 1 there are left over.  
 i.e.  $40000 = (156 \times 256) + (64 \times 1)$

Therefore we can arrive at the same numbers using the following procedure:

MSB = INT (40000/256) = 156  
 LSB = 40000 - (256 \* MSB) = 64

Either of these methods may be used to determine the LSB and MSB of any address for which they are required.

Many settings require this exact format to be used in order to change them. For example, the memory size is stored in addresses 16561 - 16562 in the format LSB,MSB.

i.e. 16561 contains the LSB, and 16562 contains the MSB.

This is true in general for all two byte quantity storers. The first address contains the LSB and the second address contains the MSB. Therefore to set a memory size of 40000 we would have to type

POKE 16561,64  
 POKE 16562,156

CLEAR xxxx

where xxxx is the string space required.

The CLEAR forces BASIC to recognize the new top of memory. This technique for setting the memory size can be used on both cassette and disk systems and saves the operator from having to enter the memory size at power-up in response to the "Memory Size?" question. The program included with this arti-

cle provides an example of the use of this facility.

POKE is also used on the model 3 to set values for a whole range of different features, e.g. to prevent the top two lines of the screen from scrolling, type  
 POKE 16916,2

or, to set the special characters mode, type  
 POKE 16420,1

(This saves using PRINT CHR\$(22) which can be a nuisance since it is effectively only a toggle switch and the programmer can never be certain which mode is set).

For other useful addresses on the Model 3, refer to the Model 3 BASIC Reference Manual, pp. 83-84.

**USR:** The USR function is used to provide an interface between a BASIC program and a machine language subroutine to be called from the BASIC program. Once a machine language subroutine has been poked into memory (see later) BASIC needs a way to call it. The USR function caters for this requirement.

Before BASIC can call a machine language subroutine, it needs to know where the entry point is located in memory, i.e. where to start executing from. In Cassette BASIC this is achieved by POKEing the address in LSB, MSB format into memory locations 16526-16527. For Disk BASIC it is achieved by typing  
 DEFUSRx = address  
 where x is a number from 0 to 9 indicating which USR routine is being used (since Disk BASIC provides the choice of 10 possible USR routines).

e.g. If the entry point is 40000,

For Cassette BASIC, type

POKE 16526,64 (LSB)

POKE 16527,156 (MSB)

For Disk BASIC, type

DEFUSRO = 40000

or DEFUSRO = &H9C40 to use USR routine 0

The machine language subroutine can then be accessed by typing

X = USR (arg) for Cassette BASIC  
 and X = USRO (arg) for Disk BASIC  
 (Note: The Disk BASIC call will function correctly without the 0 but it is always safest to include it so you don't forget which routine you're accessing).

The number enclosed by the parentheses, (arg), is an integer argument which can be sent to the machine language routine. For example, if we had a routine to scroll a certain number of lines up the screen, the argument would be the number of lines that we wanted to scroll. The argument sent to the routine may be any integer in the range -32768 to +32767 inclusive. If the programmer does not wish to send an argument then the number enclosed by the parentheses is considered to be a dummy argument and is only there to satisfy the syntax requirements of the USR function.

The variable assigned to the USR routine (in this case X) will contain the argument sent from the machine language routine, if any.

**LOADING MACHINE LANGUAGE SUBROUTINES INTO MEMORY:** There are several ways of loading machine language subroutines into memory. The easiest and the most obvious way is to load it via the SYSTEM mode in a cassette system or via the DOS command LOAD in a disk system. However, these two methods are only any good if you have an assembled version of the subroutine stored on disk or tape, whatever the case may be.

For other methods of loading machine language subroutines into memory, I thoroughly recommend a copy of Lewis Rosenfelder's book "BASIC



Faster and Better & Other Mysteries'', which is available from MICRO-80 for \$39.95 and is also listed in Tandy's RSC-9 Catalogue for \$39.95.

The method that I will use is the one which I consider to be the easiest to understand and modify for the various memory sizes. This method involves POKEing the values into memory byte by byte from DATA statements.

I have included two sample programs with this article. The first of these programs provides a substitute for the BASIC INPUT statement. It is superior to the statement it replaces in the following respects:

1. The BREAK, CLEAR and all arrow keys except the back arrow are all locked out.
2. The ENTER key will be ignored if the current length of the input is zero.
3. It provides a flashing cursor (for both Models 1 and 3).
4. The cursor may be changed to any character available in the computer's character set with a simple POKE statement.
5. It will only accept a predetermined length of input (specified by the pro-

grammer in the USR call) and then the cursor is changed to a non-flashing program-definable character indicating that no more input will be accepted.

6. Because it is written in machine language, it cannot be out-typed as can so many of the equivalent BASIC routines.
7. It shows the operator how many characters may be entered by displaying a number of characters on the screen corresponding to the maximum length of the input.
8. It will accept all delimiters (e.g. commas) without ignoring the characters which are entered after them.

In summary this routine effectively gives BASIC a super-powered LINE-INPUT function. It can be used on both cassette and disk systems, as can the BASIC program which enters the routine and demonstrates how to use it in your program.

Program Listing 1 is the documented source code for the USR routine to replace the INPUT statement. This routine only makes use of four ROM routines all of which are located in the same place on Models 1 and 3. These are

the 49H ROM routine which waits for a character to be entered from the keyboard, and the 2BH ROM routine which accepts a character from the keyboard if a key has been pressed. Both of these ROM routines are documented in the technical information section of the Model III Reference Manual. I checked with the Memory Map for Level II in '80 Micro — a Wayne Green Publication (Dec. 82 pp. 298-311) and found these same ROM routines for the Model I. The other ROM routines referred to are the ones for accepting and sending arguments from and to BASIC. These routines (0A7FH and 0A9AH) are documented under the USR function in both the Level II and Model 3 BASIC Reference Manuals. They are the same for both computers. Therefore I foresee no problems with getting this routine to work on either computer.

The source listing as shown has been assembled for a 48K computer with an origin of FFOOH. However, this should be changed to BFOOH for a 32K computer and 7FOOH for a 16K computer. The source code may then be entered into a computer via an editor/assembler such as EDTASM.

```

00010 ;*****
00020 ;*
00030 ;*          PROGRAM LISTING 1
00040 ;*
00050 ;*  USR ROUTINE TO REPLACE THE 'INPUT' STATEMENT
00060 ;*
00070 ;*          COPYRIGHT (C) 1983 BY G.S.THOMAS
00080 ;*
00090 ;*****
00100 ;

FF00      00110      ORG      0FF00H
4020      00120 CURPOS EQU      16416      ;CURSOR POSITION ADDRESS
0A7F      00130 BASARG EQU      0A7FH      ;ARGUMENT FROM BASIC
0A9A      00140 ALARG EQU      0A9AH      ;ASSEMBLY LANGUAGE ARG
0049      00150 KBWAIT EQU      49H      ;WAIT FOR CHAR FROM KB
002B      00160 KBCHAR EQU      2BH      ;GET CHARACTER FROM KB
FF00 1841 00170      JR      START      ;SKIP STORAGE AREA
0040      00180 INPUT DEFS      64      ;ROOM TO SAVE TEXT
0001      00190 CHAREN DEFS      1      ;CONTAINS CHAR ENTERED
FF43 CD7F0A 00200 START CALL      BASARG      ;GET ARG FROM BASIC
FF46 45      00210 BEGIN LD      B,L      ;B CONTAINS MAX LENGTH
FF47 0E00      00220      LD      C,0      ;C CONTAINS CURRENT LENGTH
FF49 C5      00230      PUSH     BC      ;SAVE LENGTH INFORMATION
FF4A 2A2040 00240      LD      HL,(CURPOS) ;GET CURSOR POSITION
FF4D E5      00250      PUSH     HL      ;SAVE IT
FF4E 3688      00260 PROMPT LD      (HL),136 ;PRINT CHR$(136) AT CURPOS
FF50 23      00270      INC      HL      ;NEXT SCREEN POSITION
FF51 10FB      00280      DJNZ     PROMPT ;UNTIL MAX LENGTH
FF53 3620      00290      LD      (HL),32 ;BLANK ONE CHAR AFTER
FF55 E1      00300      POP      HL      ;RETRIEVE OLD CURPOS
FF56 D1      00310      POP      DE      ;LENGTH INFO INTO DE
00320 ;

FF57 7A      00330 READKB LD      A,D      ;GET MAXIMUM LENGTH
FF58 BB      00340      CP      E      ;COMPARE CURRENT LENGTH
FF59 282C      00350      JR      Z,MAXLEN ;GO IF MAX = CURRENT
FF5B 0E02      00360 FLASH LD      C,2      ;NO. OF TIMES THROUGH LOOP
FF5D 065A      00370      LD      B,90      ;DELAY TIME
FF5F 368F      00380 CURON LD      (HL),143 ;GRAPHICS BLOCK
FF61 D5      00390      PUSH     DE      ;SAVE LENGTH INFORMATION
FF62 CD2B00 00400      CALL     KBCHAR ;GET CHAR IF AVAILABLE
FF65 D1      00410      POP      DE      ;RETRIEVE LENGTH INFO
FF66 FE00      00420      CP      0      ;KEY PRESSED ?
FF68 2024      00430      JR      NZ,KEY ;IF YES THEN GO
FF6A 10F3      00440      DJNZ     CURON ;ELSE TRY AGAIN
FF6C 0D      00450      DEC      C      ;ONE LESS LOOP
FF6D B9      00460      CP      C      ;ANY LOOPS LEFT?
FF6E 20EF      00470      JR      NZ,CURON ;IF YES THE GO
FF70 0E02      00480      LD      C,2
FF72 065A      00490      LD      B,90
FF74 3688      00500 CUROFF LD      (HL),136 ;DOT PROMPT
FF76 D5      00510      PUSH     DE
FF77 CD2B00 00520      CALL     KBCHAR
FF7A D1      00530      POP      DE

```

```

FF7B FE00      00540      CP      0
FF7D 200F      00550      JR      NZ,KEY      ;ETC
FF7F 10F3      00560      DJNZ     CUROFF
FF81 0D        00570      DEC      C
FF82 B9        00580      CP      C
FF83 20EF      00590      JR      NZ,CUROFF
FF85 18D4      00600      JR      FLASH      ;KEEP FLASHING
FF87 363C      00610      LD      (HL),60      ;PRINT "<"
FF89 D5        00620      PUSH     DE      ;SAVE LENGTH INFORMATION
FF8A CD4900     00630      CALL    KBWAIT      ;WAIT FOR CHAR FROM KB
FF8D D1        00640      POP      DE      ;RETRIEVE LENGTH INFO
              00650 ;
FF8E 3242FF     00660      LD      (CHAREN),A      ;SAVE THE CHAR ENTERED
FF91 3E00      00670      LD      A,0
FF93 BB        00680      CP      E      ;CURRENT LENGTH ZERO?
FF94 3A42FF     00690      LD      A,(CHAREN)      ;RESTORE THE CHARACTER
FF97 2811      00700      JR      Z,LETTER      ;IF E=0 THEN GO
FF99 FE0D      00710      CP      13      ;<ENTER> ?
FF9B 2831      00720      JR      Z,ENTER      ;IF YES THEN GO
FF9D FE08      00730      CP      8      ;<ERASE> ?
FF9F 281F      00740      JR      Z,ERASE      ;IF YES THEN GO
FFA1 FE18      00750      CP      24      ;<SHIFT-ERASE> ?
FFA3 2005      00760      JR      NZ,LETTER      ;IF NOT THEN GO
FFA5 6A        00770      LD      L,D      ;MAX LENGTH BACK INTO L
FFA6 2600      00780      LD      H,0      ;RESET H
FFA8 189C      00790      JR      BEGIN      ;START INPUT AGAIN
FFAA 7A        00800      LD      A,D      ;GET MAXIMUM LENGTH
FFAB BB        00810      CP      E      ;COMPARE CURRENT LENGTH
FFAC 28A9      00820      JR      Z,READKB      ;IF MAX=CURRENT THEN GO
FFAE 3A42FF     00830      LD      A,(CHAREN)      ;RESTORE THE CHARACTER
FFB1 FE20      00840      CP      32      ;LOWER ASCII LIMIT
FFB3 FA57FF     00850      JP      M,READKB      ;IGNORE CHAR IF LOWER
FFB6 FE7B      00860      CP      123      ;UPPER ASCII LIMIT
FFB8 F257FF     00870      JP      P,READKB      ;IGNORE CHAR IF GREATER
FFBB 77        00880      LD      (HL),A      ;PRINT THE CHARACTER
FFBC 1C        00890      INC      E      ;INCREMENT THE LENGTH
FFBD 23        00900      INC      HL      ;NEXT SCREEN POSITION
FFBE 1897      00910      JR      READKB      ;GO BACK
              00920 ;
FFC0 7A        00930      ERASE    LD      A,D      ;GET MAXIMUM LENGTH
FFC1 BB        00940      CP      E      ;COMPARE CURRENT LENGTH
FFC2 2006      00950      JR      NZ,NOTMAX      ;GO IF MAX <> CURRENT
FFC4 3620      00960      LD      (HL),32      ;PRINT A SPACE
FFC6 1D        00970      PROCES   DEC      E      ;DECREMENT LENGTH
FFC7 2B        00980      DEC      HL      ;PREVIOUS SCREEN POSITION
FFC8 188D      00990      JR      READKB      ;GO BACK
FFCA 3688      01000      NOTMAX   LD      (HL),136      ;PRINT DOT PROMPT
FFCC 18F8      01010      JR      PROCES      ;PROCESS THE SPECS
              01020 ;
FFCE 7A        01030      ENTER   LD      A,D      ;GET MAXIMUM LENGTH
FFCF 93        01040      SUB     E      ;SUBTRACT CURRENT LENGTH
FFD0 2004      01050      JR      NZ,CLINE      ;IF DIFFERENCE<>0 THEN GO
FFD2 3620      01060      LD      (HL),32      ;PRINT A SPACE
FFD4 1806      01070      JR      BASIC      ;BACK TO BASIC
FFD6 47        01080      CLINE   LD      B,A      ;DIFFERENCE INTO B
FFD7 3620      01090      BLANK   LD      (HL),32      ;BLANK OUT SCREEN POSITION
FFD9 23        01100      INC      HL      ;NEXT SCREEN POSITION
FFDA 10FB      01110      DJNZ     BLANK      ;UNTIL ALL DONE
FFDC 4B        01120      BASIC   LD      C,E      ;CURRENT LENGTH INTO C
FFDD D5        01130      PUSH     DE      ;SAVE LENGTH INFORMATION
FFDE 2A2040     01140      LD      HL,(CURPOS)      ;GET ORIGINAL CURPOS
FFE1 1102FF     01150      LD      DE,INPUT      ;DESTINATION
FFE4 0600      01160      LD      B,0      ;RESET B
FFE6 EDB0      01170      LDIR      ;SAVE THE TEXT
FFE8 D1        01180      POP      DE      ;RETRIEVE LENGTH INFO
FFE9 2600      01190      LD      H,0      ;RESET H
FFEB 6B        01200      LD      L,E      ;FINAL LENGTH INTO L
FFEC C39A0A     01210      JP      ALARG      ;PASS IT TO BASIC
0000          01220      END

```

For those who do not have an editor/assembler, Program Listing 2 is the BASIC program which will enter the routine into memory. This program will work on either a disk system or a cassette system with any memory size. The numbers contained in the DATA statements are the decimal equivalents of the hexadecimal numbers in the second column on the left of the source listing.

```

10 *****
20 *
30 *          PROGRAM LISTING 2
40 *
50 * BASIC PROGRAM TO POKE USR ROUTINE INTO
60 * MEMORY TO REPLACE THE INPUT STATEMENT
70 *
80 *          COPYRIGHT (C) 1983 BY G.S.THOMAS
90 *
100 *****

```



```

110 '
120 'NOTE: The routine will work as is without changing any
130 ' of the settings. The settings are there purely and
140 ' simply for the programmer's convenience.
150 '
160 ' To save memory, all the REM statements can be removed
170 ' and the smaller lines can be compounded together using
180 ' the colon (":").
190 ' For example, see line 360
200 '
210 DATA24,65,205
220 DATA127,10,69,14,0,197,42,32,64,229,54,136,35,16,251,54,32
230 DATA225,209,122,187,40,44,14,2,6,90,54,143,213,205,43,0,209
240 DATA254,0,32,36,16,243,13,185,32,239,14,2,6,90,54,136,213
250 DATA205,43,0,209,254,0,32,15,16,243,13,185,32,239,24,212,54
260 DATA60,213,205,73,0,209,50,66,255,62,0,187,58,66,255,40
270 DATA17,254,13,40,49,254,8,40,31,254,24,32,5,106,38,0,24,156
280 DATA122,187,40,169,58,66,255,254,32,250,87,255,254,123,242
290 DATA87,255,119,28,35,24,151,122,187,32,6,54,32,29,43,24
300 DATA141,54,136,24,248,122,147,32,4,54,32,24,6,71,54,32,35
310 DATA16,251,75,213,42,32,64,17,2,255,6,0,237,176,209,38
320 DATA0,107,195,154,10
330 TM=PEEK(16561)+PEEK(16562)*256 'Get Top of Memory
340 N2=INT(TM/256) 'Calculate Most Significant Byte
350 IFTM=256*N2<172THENN2=N2-1 'Make room for routine
360 POKE16561,255:POKE16562,N2-1 'Set NEW Top of Memory
370 CLEAR50 'Make BASIC recognize new Top of Memory
380 NT=PEEK(16562)+1 'MSB of Start of Routine
390 SA=NT*256 'Starting Address for Routine
400 IFSA>32767THENSA=SA-65536 'Prevent OVERFLOW Error
410 READA:POKESA,A:READA:POKESA+1,A 'POKE first two bytes
420 FORI=0TO171 '174 bytes in the DATA statements
430 READA 'Read them
440 IFA=255THENSA=NT 'Change the NON-RELOCATABLE instructions
450 POKESA+I+67,A 'Put the byte into memory
460 NEXTI 'Do the next one
470 'Point to the USR routine:
480 ONERRORGOTO3020:DEFUSR=SA:GOTO500 'DISK Systems
490 POKE16526,0:POKE16527,NT 'CASSETTE Systems
500 MS=NT*256 'Reference address as stated in article
510 IFMS>32767THENMS=MS-65536 'Adjust if necessary
1000 '
Your program starts here
1010 'This is a demonstration program
1020 CLS:PRINT"What is your name? "; 'Question
1030 L=20 'Maximum permitted length
1040 POKEMS+115,45 'Halve period for which cursor is "off"
1050 POKEMS+136,191 'CHR$(191) at end of input
1060 POKEMS+117,42 'Set cursor "off" character to "*"
1070 GOSUB2000 'Call subroutine to accept input
1080 N$=I$ 'Save contents of input - I$ will be wiped next time
1090 PRINT:PRINT"Your name is "N$".
1100 PRINT"How old are you, "N$"? "; 'New question
1110 L=2 'Maximum length
1120 POKEMS+96,35 'Set cursor "on" character to "#"
1130 POKEMS+136,60 'End of input char to "<"
1140 POKEMS+94,180 'Double the original "off" delay
1150 POKEMS+117,32 'Set cursor "off" character to " "
1160 POKEMS+79,46:POKEMS+203,46 'Length of input char to "."
1170 GOSUB2000 'Call subroutine to accept input
1180 AGE=VAL(I$) 'Save numeric value of I$
1190 PRINT:PRINTN$ " is"AGE"years old."
1200 END 'End of Program
1999 '
Subroutine to accept input from the keyboard and set up
I$ to point to this input.
2000 I$="" 'Clear the variable to contain the input
2010 X=USR(L) 'Call routine - set max length of L
2020 POKEVARPTR(I$),X 'Set length of I$
2030 POKEVARPTR(I$)+1,2 'LSB of address to I$
2040 POKEVARPTR(I$)+2,NT 'MSB of address to I$
2050 RETURN 'Return from the subroutine
3000 '
Error trap to catch BASICs which have no DEFUSR statement
3010 'Resume execution only if error is in line 430
3020 IFERL=480THENRESUME490ELSEONERRORGOTO0:END

```

The program operates in the following manner:

It obtains the current top of memory from the addresses 16561-16562 and allocates room for itself just below the top of memory. This is done in 256 byte increments so as to make relocating the routine easy. Unfortunately, I could not make the routine truly relocatable (I had to include two JP instructions which are not relocatable) and therefore I had to have the BASIC program which POKEd the routine into memory do the relocating for me. It then resets the top of memory to protect itself and prevent BASIC from storing data up there and writing over the routine. It then proceeds to POKE the data into memory and once this has been completed, the routine is ready for use. The program then sets up the USR routine pointers using the addresses 16526-16527 or the DEFUSR statement, whichever is applicable.

To access the routine from BASIC, insert a line of the following form in your program:

```

1000 I$= " " 'Clear the variable to
                        contain the input
:X = USR(L) 'Make the call
:POKE VARPTR(I$),X 'Set the length
                        of I$
:POKE VARPTR(I$)+1,2'LSB of address
                        to I$
:POKE VARPTR(I$)+2,NT 'MSB of
address to I$

```

The argument L is the maximum length of input to be enforced by the routine. It may be a variable, a constant or an expression. Therefore if L = 15 then the routine will only accept 15 legal characters of input from the keyboard. On return from the routine the actual number of characters entered by the operator is stored in the variable X.

This demonstrates a very powerful use of the POKE statement in conjunction with the VARPTR (or variable pointer) function. If you have a string variable, for example A\$, and you type PRINT VARPTR(A\$)

the address where the length of the string contained in A\$ is stored in memory will be returned. Therefore it follows that if you type

```
PRINT PEEK(VARPTR(A$))
```

then the length of the string contained in A\$ will be returned. Similarly,

```
PRINT PEEK(VARPTR(A$)+1)
```

will return the LSB of the starting address of the contents of A\$ in memory and

```
PRINT PEEK(VARPTR(A$)+2)
```

will return the MSB of the same address.

Proceeding further on this idea, if you let

```
F = PEEK(VARPTR(A$)+1) +
```

```
PEEK(VARPTR(A$)+2) * 256
```

and then type

```
PRINT PEEK(F)
```

then the ASCII value of the first character in A\$ will be displayed.

The program uses this idea in the last three POKE statements. Since I\$ is the variable set up to contain the input and X is the actual number of characters entered by the operator, it seems reasonable that the length of I\$ should be X. This is the purpose of the first POKE statement. It merely POKES X in the memory location where the length of I\$ is stored. The purpose of the next two POKES is to point I\$ to the input entered by the operator which is located in a known buffer set up in the machine language routine. The BASIC program to enter the routine is set up in such a way as to always have the LSB of the address containing the input as 2. The MSB of the address is the MSB of the memory size + 1 (since the input is stored in the address which is 2 bytes above the max-

imum memory address accessible by BASIC i.e. the memory size). The variable NT contains the MSB of the memory size and it is this value which is POKEd into the MSB location. On completion of these three POKES, the variable I\$ will contain the input entered by the operator and may be processed in the same manner as any standard string variable (because that's all it is). To process numeric data, the VAL function of BASIC can be used. e.g. if \$1 = "23" then PRINT VAL(I\$) will return the number 23 as distinct from the string "23".

The routine should be POKEd into memory very early on in your program since it adjusts the memory size and clears all variables on initialization. It only uses about 250 bytes of top memory and once it has been entered into memory the data lines can be deleted if memory is precious.

In order to be able to make use of the many features of the routine, it is advantageous to set up a variable just after the initialization sequence as follows:

MS = NT \* 256

Once this has been done POKing different numbers into the following addresses allows you to change the settings of the routine:

SETTING	ADDRESS	DEFAULT
Character to be displayed indicating length of input	MS + 79, MS + 203	136
Cursor character (when "on")	MS + 96	143
Cursor character (when "off")	MS + 117	32
Character to be displayed indicating end of input	MS + 136	60
Cursor flashing speed—"on" delay	MS + 94	90
"off" delay	MS + 115	90

All settings listed here require integer values in the range 0-255. In the case of characters to be displayed, ensure that the number you choose has a character corresponding to it which is displayable by your computer. Where two addresses are listed, this means that both addresses must be changed for the feature to function properly.

There are two states in which the cursor can be. It can either be "on" (like it is for the length of time that you can see it while it is flashing) or it can be "off". If these states change at regular time intervals (e.g. every half a second) the effect of a flashing cursor is created. This is what is meant by the cursor being "on" and "off". We usually associate a flashing cursor with a CHR\$(143) graphics block blinking on and off. In this case, the "on" character is CHR\$(143), the graphics block, and the "off" character is a CHR\$(32), a blank. This routine allows you to define what the cursor's "on" and "off" characters are and by POKing values into the specified addresses you can change these characters to what every you like.

By POKing different values into the addresses specified above for the "on" and "off" delay, you can increase or decrease the flashing speed of the cursor according to your requirements. To see which values are currently set, all you have to do is PEEK the addresses corresponding to the feature about which you require information and this will tell you the value of the setting. Always remember that just because you have a machine language routine stored up there in protected memory, that doesn't mean that you cannot modify its operation to suit your own ends by using the POKE statement.

In conclusion, I hope that this article has fulfilled the needs of those people who requested it, and since this is my first article, I would appreciate any feed-

back on it (whether positive or negative) so that I will know whether I have covered the requested material adequately. 58 Warnbro Beach Road, SAFETY BAY, W.A. 6169

I will be happy to try and answer any queries that readers may have. Please include a self-addressed stamped envelope.

# THE ADVENTURE SYSTEM

## A SOFTWARE REVIEW

by Ed Grigonis

Those of you who have ever considered writing an Adventure would know that there are a number of ways of achieving this objective. These various methods may be summarised as follows:—

(a) Dig into an existing Adventure and change the existing data. The major flaw with this method is that you are restricted to the original Adventure format.

(b) Write a BASIC Adventure. This is OK but you will find that memory limitations (particularly in a 16K computer) will prove frustratingly restrictive. The process of actually coding the Adventure will also detract from the task in hand.

(c) Use the Adventure Generator included in "The Captain 80 Book of Basic Adventures" to get rid of the drudgery. This is OK as well, but you may still be hampered by memory limitations.

(d) Learn machine language. If you want the latest fantastic graphics Adventure then this is the way to go. Be prepared to devote a lot of time to the task!

If, on the other hand, your only objective is to write a great Adventure and you aren't particularly interested in graphics then you can always take the easy way out. This will still involve learning a new computer language but the effort required is minimal when stacked up against the alternatives.

What I am talking about is the Adventure Language as contained in "The Adventure System" from the Alternative Source.

First, a bit of history. Hands up all those who think Scott Adams actually sat down and wrote a completely new machine language program for each of his Adventures? Sorry, folks! 'Taint so!!! What he actually did do, very early on, was to sit down and write himself an Adventure Editor in machine language. Most of his Adventures were actually written by feeding data into this program and letting it do all the hack work. I suspect that similar methods were used to get his Adventures onto other computers.

Unfortunately, the Adventure Editor used by Scott Adams has never been available via the commercial market. Into the picture stepped Allan Moluf, well known author of programs for the TRS-80, and Bruce Hansen, author of "Tasmon". Allan Moluf produced a BASIC Adventure Editor as well as a companion BASIC Driver program to use the generated data base. Subsequent machine language

enhancements were added by Bruce Hansen. This first effort evolved into what became known as "The Adventure System".

Current versions of "The Adventure System" have been fully implemented in machine language by Bruce Hansen and are, so far, fully compatible with the Scott Adams effort. "The Adventure System" is available to anyone who cares to send The Alternative Source enough money to buy it.

I mentioned that this system requires you to learn a new language. What follows is a discussion of the language and how it is used to create an Adventure which looks just like a Scott Adams' original.

"The Adventure System" requires you to specify the following details when entering your data:

- Objects.
- Messages.
- Rooms.
- Vocabulary.
- Actions.
- Header information.

## OBJECTS

Each object specified in the data must contain three parameters: object number, starting location and object description.

The object number is specified by the Adventure Editor when you first enter the information. The first object will always be numbered 0 (zero) and the last object will always be numbered one less than the total number of objects.

The starting location is the number of the room in which the object will be placed at the beginning of the Adventure, i.e. if you specify 3 then the object would initially be located in Room 3.

The object description tells you what the object actually is, whether it can be manipulated and whether or not it is a treasure.

Consider the following three objects:

0: 3 Pile of rocks

1: 5 Large rock/ROCK/

2: 4 \*SPARKLING DIAMOND\*/DIAM/

Object number zero will be found when the adventurer goes into Room 3 with the above description displayed by the Adventure Driver. The object cannot be manipulated.

Object number one will be found in Room 5. The provision of the short description between slashes tells the Adventure Driver that this object may be manipulated, i.e. picked up, thrown, whatever.

Object number two will be found in Room 4. This object can be manipulated. The leading asterisk in the object description tells the Adventure Driver that this object is a Treasure. Object two would be taken into account for scoring purposes if you placed it in the Treasure Room.

As you can see, there is nothing difficult about specifying objects.

Object number 13 is reserved for the artificial light source, viz. lamp, matches, etc.

## MESSAGES

Specifying messages is even simpler than objects. The Adventure Editor will specify suitable numbers when you first enter the messages. Message 0 is reserved so your messages will always start at number one.

A sample message would appear as follows:

1: There is a fly in my soup.

This message would be printed



whenever the driver was required to print message number one.

## ROOMS

The parameters required when entering Room details into the Editor are Room Number (given by the Editor), allowable directions and Room description.

When entering Room details the Editor will ask for six Room numbers corresponding with Rooms which may be entered from the current Room. These numbers will also correspond with the directions North, South, East, West, Up and Down.

The Room description determines what the Driver will print when you are in a particular Room.

Consider the following two Room descriptions:

1: 0 5 23 0 0 0 large crate  
2: 0 0 0 12 6 0 \*I'm outside the shop

Room number one leads into Rooms 5 and 23 by going South and East respectively. The Driver would show South and East as obvious when you are in Room 1. The actual Room description would be preceded by the phrase "You're in a".

Room number two leads into Rooms 12 and 6 by going West and Up respectively. The leading asterisk in the description prevents the default phrase being printed.

It is important to note that the allowable directions can be overridden by allowing the adventurer to move to another room by specifying a suitable command, i.e. GO SHED.

If the adventurer moves in an illegal direction when a light source is available, he will receive the message "I can't go in that direction". If he moves in an illegal direction in the dark, he will be killed.

The last Room is reserved for a "Limbo" state when the person has been killed, and may or may not have exits into other rooms.

## VOCABULARY

The vocabulary details all verbs and nouns which may be used in the Adventure.

Predefined Verbs are as follows:

0 AUTO  
1 GO  
10 GET  
18 DROP

Predefined Nouns are:

0 ANY  
1 NORTH  
2 SOUTH  
3 EAST  
4 WEST  
5 UP  
6 DOWN

Primary vocabulary words may have as many synonyms as required but these must follow the relevant primary word and must each be preceded by an asterisk, i.e.

7 DOG  
8 \*HOUND  
9 \*BASSET

## ACTIONS

Actions are the heart of an Adventure and are where newcomers to "The Adventure System" will have the most difficulty.

Action entries contain the following information:

Verb, Noun, Conditions, Commands and Action Titles.

Action Titles simply document the function of a particular entry. They act as comments and may be omitted, although as with any programming language, it is

a good idea to liberally comment source code.

The verb and noun entered by the adventurer are used to determine which conditions and commands will be acted on. For example, if the adventurer were to enter "CLIMB TREE", the Driver would only consider those Action entries with the verb, noun combination of "CLIMB TREE". Of course, if "CYPRESS" had been defined as a synonym of "TREE", then "CLIMB CYPRESS" would also be acceptable in the example given.

The "conditions" of an Action entry provide a list of test which must be passed in order for the "commands" of the Action entry to be carried out. The "commands" of the Action entry are only carried out if all the "conditions" are met.

Before describing a few Action entries I will discuss the available "conditions" and "commands".

## CONDITIONS

PAR This condition always passes and is used to pass a parameter on to a command. The analogy with BASIC is the 'DATA' statement.

HAS This condition passes if the adventurer is carrying the object referred to, i.e. HAS 15.

IN/W passes if the adventurer is in the same Room as the numbered object, i.e. IN/W 15.

AVL passes if the adventurer is in the same Room as the object or is carrying the object.

IN passes if the adventurer is in the numbered Room, i.e. IN 5.

—IN/W passes if the numbered object is either held by the player or is in another ROOM, i.e. —IN/W 15

—HAVE passes if the player is not carrying the numbered object.

—IN passes if the player is not in the numbered Room.

BIT passes if the numbered bit flag is set.

—BIT passes if the numbered bit flag is cleared.

ANY passes if any objects are being carried.

—ANY passes if no objects are being carried.

—AVL passes if the numbered object is in any other Room.

—RMO passes if the numbered object is not in Room zero.

RMO passes if the numbered object is in Room zero.

CT < = passes if the counter is less than or equal to the number specified.

ORIG passes if the numbered object is in the same room it started in.

—ORIG passes if the numbered object is not in the same Room it started in.

CT = passes if the counter is equal to the number specified.

(Any numbers input with a condition must be in the range 0-1600)

## COMMANDS

0 This is a "null" command.

1-99 Display messages 1-99.

GETX If this followed a PAR 5, then object number 5 would be picked up.

DROPX If this followed a PAR 5, then object number 5 would be dropped.

GOTOY If PAR 22 was specified, the player would be moved to Room 22.

X-RMO PAR 5 would cause object 5 to be moved to Room zero.

NIGHT Sets the light/darkness bit flag (15). If the artificial light source is unavailable the Room will be dark and no Room description will be given.

DAY Clears the light/darkness flag.  
SETZ PAR 4 would cause bit flag number 4 to be set.

X—>RMO PAR 5 would cause object 5 to be moved to Room zero.

CLRZ PAR 4 would cause bit flag number 4 to be cleared.

DEAD Clears the light/darkness flag, moves the player to the last Room and tells him he is dead.

X—>Y PAR 5 PAR 22 would cause object number 5 to be moved to Room number 22.

FINI Indicate that the game is over and enquire about a replay.

DSPRM Display current room if it is light or the artificial light source is present, else display "It's too dark to see".

SCORE Display number of treasures in the Treasure Room and the percentage of treasures stored.

INV List inventory.

SETO Set bit flag zero.

CLRO Clear big flag zero.

FILL Refill the artificial light source.

CLS Has no effect but was included to maintain compatibility with the original BASIC system.

SAVE Save the game to disk or tape depending on the version.

EXX,X PAR 5 PAR 15 would cause the location of object number 5 to be swapped with the location of the object number 15.

CONT Allows continuation of an Action entry.

AGETX PAR 5 would enable object number 5 to be obtained even if the carry limit has been exceeded.

BYX—>X PAR 5 PAR 15 would cause object number 5 to be placed in the same Room as object number 15.

CT-1 Subtract one from the counter value.

DSPCT Display current value of the counter.

CT<—N PAR 100 would set the counter to a value of 100.

EXRMO Exchange the current Room number with the Room number held in Alternate Room Register zero.

EXM,CT PAR 5 would cause the value of the current counter to be exchanged with counter number 5.

CT+ PAR 60 would add 60 to the current counter.

CT-N PAR 10 would subtract 10 from the current counter.

SAYW Display the noun (second word) input by the player.

SAYCR Start a new line on the display.

EXC,CR PAR 2 would cause the current Room number to be swapped for the Room number currently held in Alternate Room Register number 2.

DELAY Pause for about 1 second before going on to next command.

## EXAMPLES

To discuss all of the possibilities inherent in the above would take a year's issues of this magazine so I will just give a few examples from the Manual.

0: AUTO 100 —BIT 1 PAR 1 0 0 0  
MSG 1 SETZ — — INTRO

The 0: shows that this is Action 0. AUTO 100 causes this action to be considered all of the time. When the Adventure is started, all bit flags are clear. In this case —BIT 1 would therefore be true. PAR 1 passes the parameter 1 to the

commands. The 0's indicate that there are no more conditions to be met. MSG 1 causes Message Number 1 to be printed. SETZ obtains the parameter of 1 from the PAR 1 condition and therefore sets bit flag number 1. Unless bit flag number 1 is cleared at a later stage by a different command, this is the last time this command will be executed. The dashes indicate that there are no more commands in the Action entry. INTRO is simply a comment.

```
11: GET KEY IN/W 12 PAR 12
0 0 0 GET MSG 5 — —
```

If the player were to enter "GET KEY" this Action entry would be considered. If the player is in the same Room as the key (IN/W 12) the parameter of 12 (presumably the object number of the key) would be passed to the GETX command which would cause the player to pick up the key. Message 5 would then be printed. Note that if the player was already carrying the key he would automatically get the message "I'm already carrying it".

```
14: GO DOOR IN 2 PAR 3 0 0
GOTOY MSG 5 — —
```

If the player was to enter "GO DOOR" then, provided he was currently in Room 2, the parameter of 3 would be passed to the GOTOY command and the player would be moved to Room number 3. Message number 5 would be printed. These are only three examples.

However, by studying these examples and also the conditions and commands summarised earlier, you will see that the scope provided within "The Adventure System" is indeed comprehensive.

#### HEADERS

The best way to explain the header is to give an example from the manual.

Adventure Z Version 1.01 14500 bytes free

```
Bytes under 16K = 7523
#OBJ #ACT #VOC #RM
14 41 22 8
MAX BEG #TR WLEN
5 1 1 4
TIME #MSG TR-RM
999 16 7
```

The name of the Adventure is "Adventure Z" and the version number is 1.01. There are currently 14,500 bytes free, although for any particular Adventure this would depend on your computer's configuration and the presence of any high memory drivers. There would be 7523 bytes free if this Adventure was entered as a SYSTEM tape in a 16K computer. There are 14 objects and 41 actions. There are 22 verbs and 22 nouns (although one might be less). There are 8 Rooms. The adventurer can carry a maximum of 5 objects. The adventurer will begin in Room number 1. There is only one treasure. The number of significant letters in nouns and verbs is four. The time limit is 999 moves. There are 16 messages. The treasure room is Room number 7.

All of this information is entered when you commence compiling the Adventure.

#### ADV EDT LIMITATIONS

The following limitations (if you can call them that) are imposed on the data entered into the ADV EDT program:

- (1) Maximum of 500 Action entries.
- (2) Maximum of 150 vocabulary entries (150 verbs and 150 nouns).
- (3) Maximum of 100 rooms.
- (4) Maximum of 99 messages.
- (5) Maximum of 250 objects.
- (6) Maximum characters in description of object, room or message is 255.
- (7) Maximum word length of vocabulary words is 7 characters.

- (8) Maximum length of Action titles is 20 characters.

To give you a comparison, the following are the maximums from the Scott Adams Adventures:

- 270 Action entries.
- 80 Vocabulary words.
- 100 Objects.

#### WHERE DAY AGED DIT???

As far as I am aware, "The Adventure System" is only available from The Alternate Source, 704 N. Pennsylvania, LANSING, MI. 48906 U.S.A. The cost of the program is \$US49.95 plus postage. If you write to them to ask about the cost of airmail, make sure you enclose some international reply coupons.

#### WADDAY AGET???

Before I tell you what you get, I should point out that you will need to have 48K of RAM available. There are versions, however, for both disk and tape.

If you order the tape version you will get two programs. ADV EDT — the actual Adventure Editor and ADV TT which will enable you to create a SYSTEM tape from any Adventure data base you create. The disk version also includes ADV which is a disk based Adventure driver. ADV EDT in both versions contains ADV so when you are writing Adventures you can jump back and forth between the Editor and Driver.

Each version also includes a short Adventure which is fully explained in the documentation as well as two full length Adventures.

The documentation is supplied in a sturdy black vinyl binder. The original manual was somewhat hard to follow in parts. However, there is now a much more detailed manual which just about ranks alongside Bruce Hansen's excellent "TASMOM" documentation for ease of use and clarity.

#### WHAT? IS THERE MORE?

Also included is Issue No. 1 of the Auggies newsletter "Augment". If you buy "The Adventure System" you are entitled to join the Auggies (Adventure User Group). For \$US12.00 per year (extra for airmail) you receive the "Augment" quarterly which includes at least one Adventure. You are eligible to market your Adventures through The Alternate Source, provided they are suitable, and thereby derive royalties. You can also purchase any Adventures created with "The Adventure System" at a discount.

The Alternate Source also have the following Adventure utilities available for purchase. ADV TAPE is similar to ADV TT and will create a SYSTEM tape from a disk data base. This avoids having to save the data base to tape first. ADV COPY will take the Adventure off a protected Scott Adams Adventure disk and place it on an unprotected disk. ADV DUMP will read in a tape Adventure and dump the data base to disk. You have to be an Auggie to get these utilities.

I could quite easily go for a lot longer. There is so much that I haven't even touched on.

To summarise, if you like playing Adventures or if you are at all interested in writing Adventures, then you should definitely buy "The Adventure System". You won't regret it!

# A REVIEW OF THE TRS-80 MC-10

(COLOUR COMPUTER)

by Charlie Bartlett

It never ceases to amaze me how computers are getting smaller all the time. When the Model 1 TRS-80 came out, it was thought to be incredible that a computer was actually crammed underneath the keyboard. Now we have the TRS-80 MC-10 upon us and it makes the Model 1 TRS-80 look like an Elephant or maybe a Dinosaur would be more appropriate considering that the Model 1 is out of production.

The MC-10 is 2 inches high, 8½ inches long and 7 inches wide, (51mm x 216mm x 178mm), and weighs in at 29½ ozs (836.32 grams). The Microprocessor is an 6803 and is not fussy about spaces around keywords as is the 6809E in the Colour Computer, which is just as well considering the 3142 bytes of memory available in the MC-10. I was quite surprised at the number of commands that such a small machine supports, in fact it has NEARLY as many commands as the unextended Basic in the Colour Computer. After some digging around in the memory I also found some commands that it supports that are NOT in the manual, more on this in a minute. Listed below are commands that are available, (as listed in the manual).

ABS	Computes absolute value.
ASC	Returns ASCII code of first character of string.
CHR\$	Returns character for ASCII or graphics code.
CLEAR	Reserves bytes of string storage.
CLOAD	Loads Basic program from cassette.
CLOAD*	Loads numeric data into an array from cassette.
CLS(x)	Clears display to specified colour "x".
CONT	Continues program execution if BREAK has been pressed.
COS	Returns cosine.
CSAVE	Saves a Basic program to tape.
CSAVE*	Saves contents of a numeric array to cassette.
DATA	Stores data in your program.
DIM	Dimensions an array.
END	Ends program.
EXP	Returns natural exponential
I	Exponentiation character.
FOR	: Creates
TO	: a loop
NEXT	: in a program.
STEP	: with step to increment.
GOSUB	Sends computer to subroutine.
GOTO	Sends computer to a line.
IF/THEN	Test a relationship.
INKEY\$	Strobes the keyboard and returns the key being pressed.
INPUT	Computer waits for input from the keyboard.
INT	Converts a number to an integer.



**LEFT\$** Returns left portion of string.  
**LEN** Returns the number of characters in a string.  
**LET** Assigns value to variable (optional).  
**LIST** Lists program lines on screen.  
**LLIST** Lists program lines to printer.  
**LOG** Returns natural logarithm.  
**LPRINT** Prints an item on the printer.  
**MEM** Returns the amount of free memory.  
**MID\$** Returns a substring of another string.  
**NEW** Erases memory contents.  
**ON × GOSUB** Multi-way branch to specified subroutines.  
**ON × GOTO** Multi-way branch to specified lines.  
**PEEK** Returns contents of a memory location.  
**POKE** Puts value into specified RAM location.  
**POINT** Tests whether a graphic cell is on or off.  
**PRINT** Prints to screen, abbreviation of ? is available.  
**TAB** Moves cursor to specified column.  
**PRINT @** Print at specified location.  
**READ** Reads the next item in a DATA line.  
**REM** Remark.  
**RESET** Erase dot that was SET.  
**RESTORE** Resets DATA pointer.  
**RETURN** Returns computer from a subroutine.  
**RIGHT\$** Returns right portion of string.  
**RND** Return pseudo random number.  
**RUN** Executive a program.  
**SET** Sets a dot at a specified location to a specified colour.  
**SGN** Return sign of specified number.  
**SKIPF** Skips to end of next program on cassette.  
**SIN** Returns sine.  
**SOUND** Sound specified tone for specified duration.  
**STOP** Stops execution of program.  
**STR\$** Converts a number to a string.  
**SQR** Returns the square root of a number.  
**TAN** Returns tangent.  
**VAL** Converts a string to a number.

Quite a powerful Basic for its size isn't it, the following command are not listed in the manual or anywhere else for that matter.

**VARPIR** This command has the normal syntax of A = VARPTR (B\$) and it works, so I don't know why it was not mentioned.

**USR** This command is supported by the Basic interpreter though at the moment it is not much use until some technical information becomes available to find out where to poke the entry points.

**EXEC** This command also has the normal syntax of EXEC addr where "addr" is the entry point of a machine language program or subroutine.

**CLOADM** This command will load a machine language program into memory and has the syntax of CLOADM "filespec" or CLOADM "filespec", addr where "addr" is the position in memory that the routine is to be loaded to.

As far as I could determine the command CSAVEM is NOT supported, (that is unless it has a very different syntax from normal), I tried several variations on the syntax and nothing worked. If that

is the case it would seem that Tandy have been very crafty, you can load a machine language tape but you cannot save one. Now you ask, if that is the case how did I determine that CLOADM works, since there are no machine language programs for the MC-10 at this time.

Well for a start CLOADM does not return a syntax error, it switches on the cassette, clears the screen and prints an "S" at the top of the screen indicating that it is searching. If a BASIC program is encountered it returns an FM error (file mode). If a machine language tape from the Colour Computer is encountered it happily loads in the file name and only fails with an I/O error when the 6809E code is encountered. Which leads me to believe that given a MC-10 machine language file, it would load.

What use are VARPTR, EXEC and USR, well even if you cannot save a pure machine language program, there is no reason why you could not have a BASIC program with machine language poked into the lines with the help of VARPTR or you could have your machine language in DATA statements and poke this into memory and then EXEC to execute.

Video Memory is from 16384 to 16895, other addresses also work in tandem, the reason for this is not understood. Some keyboard addresses are in the region of 16940 and up, for example the address 16952 is used by the space bar and returns a value of 247 when the space bar is pressed, if you write a program where you want something to happen — FOR AS LONG AS the key is held instead of IF the key is pressed then you can read this memory location for the value and if the value is found poke zero into it so that on the next pass, if the key is still held down it will return the original value and if not it will return zero, this procedure applies to the other memory locations in this area.

Blindly poking into various memory locations brought interesting responses that would seem to indicate that with some detailed information higher resolution graphics might be possible. Try poking location 20 (decimal) with different values to see what I mean, (it locks the machine up though and you will have to press the reset button).

Having a Colour Computer here, I thought I could save some typing time by loading a Colour Computer Basic program, I thought I was on to a good thing for a while since the Colour Computer and the MC-10 will both load each other's tapes. However, unfortunately different values are used for the keywords in both machines so what is loaded in has the same line numbers but all of the keywords are swapped around, pity!!

The MC-10 has a DIN socket for a cassette, no cassette cable is supplied, but if you have a Model 1 or a Colour Computer you can use the same cable as

the pin connections are the same. It also has an RS232 DIN socket, this also has the same pin configuration as the Colour Computer.

To sum up, it is a very nice little machine at a reasonable price, the only thing lacking at the moment seems to be the software.

## INPUT/OUTPUT

FROM: R.J. Mclean-Formartin, Qld.

I own a System-80 on which I am using a Tandy green screen monitor. At times the display shivers and this makes it difficult to read. However, if a key is pressed (a character entered), or the NEWLINE key is depressed the shivering stops. Would you have any idea of a solution to this problem?

(It sounds like you have an intermittent fault of some sort and you should refer the problem to a qualified technician for repair. — Ed.)

FROM: Mr. G. Whitcher—Yunderup, W.A.

It would be helpful if programs that use USR calls were printed with the necessary information to convert from Level 2 BASIC to Disk BASIC for those of us who are too mean to buy the Disk subscription for the sake of a few programs which may be of interest. I only subscribe to the cassette subscription for the ease and not having to type the programs that interest me.

(Your point is well taken and we would like to include the information needed to modify Level 2 programs with USR routines so that they will work with Disk BASIC but we do not have the time to dissect machine language routines that are only provided in object format or the BASIC equivalent. If the program author would supply commented assembly language source code for the USR routine, we would be happy to publish that along with the program listing and any relevant suggestions for disk users.

RECALL on the disk version of SOFTPAK is intended to overcome this difficulty in most cases by running the Level 2 program in the Level 2 environment. However, there are those cases where modifying the program for disk BASIC is the preferred solution.—Ed.)

FROM: Mr. J. Linton—Malabar, N.S.W.

As the SYSTEM 80 is going out of production, is there an alternative expansion system to allow update to disk drive or do I have to buy a SYSTEM 80 expansion unit before they go out of production?

(By now you will probably know the answer to this question as it is discussed in the Editorial. Briefly, MICRO-80 will have expansion interfaces for the System 80 available about the end of February, 1984. — Ed.)

## SOFTWARE

### AUSTRALIA'S CUP —L2/16K

by Carl Cranstone

Australia recently won the America's Cup which the Americans had held for 132 years. The Australian yacht

"Australia II" broke the longest winning-streak in sporting history by defeating the American yacht "Liberty" four wins to three. Every Australian can now relive that day (or should I say, early morning!) by playing the Australia's Cup game!!! Up to six people can play and place bets on the outcome of the races. Complete instructions on how to play are included in the program.

### HOW TO GET THE PROGRAM RUNNING(?)

To get the program 'Australia's Cup' running, those of you who are entering the

program from the magazine will have to follow my instructions below to the letter (or at least to the word!). There are three programs that I have written for this game. The first AUCUP/LST is the Australia's Cup program in its raw form (i.e. it contains no graphics). The second is AUCUP/DAT which is a program that creates the program lines that contain the graphics. The third is AUCUP/LNW which is an LNWBASIC Program to enable LNWB-80 owners to see Australia's Cup in colour.

### 1. AUCUP/DAT

This program reads in the data for the graphics and POKEs it into memory — creating assembled strings. (Exactly the same as those created by Charlie Bartlett's Graphic Assembler program — issue 5 April 1980).

\*\*\*STEP ONE: Type in the AUCUP/DAT program and RUN it

\*\*\*STEP TWO: delete lines 1 & 2

\*\*\*type LIST— you should see something resembling a garbaged program. Don't worry! This garbage is really assembled strings.

\*\*\*SAVE or CSAVE the strings immediately!!!

### 2. AUCUP/LST

This is the program which uses the strings.

\*\*\*STEP THREE: LOAD OR CLOAD the strings and type in this program around the strings. You will notice that lines 1350 and 1560 in this program are reserved for the strings.

\*\*\*STEP FOUR: When you have typed in the program around the strings SAVE or CSAVE it immediately. (This program will now be known as AUCUP/BAS).

WARNING: BE VERY CAREFUL THAT YOU DO NOT ACCIDENTALLY ERASE A LINE WITH AN ASSEMBLED STRING IN IT OR EDIT IT. THIS CAN RUIN ALL OF YOUR HARD WORK!!!

\*\*\*STEP FIVE: LOAD or CLOAD the program in and RUN it. The first thing you should see is a large map of Australia. You must press ENTER to continue into the game. If you don't, the game will go into Demo Mode. To start the game you must press ENTER from the map. If you press ENTER from the Demo Mode, you will return to the map.

### 3. AUCUP/LNW

This is for the benefit of LNWB-80 owners who want to see Australia's Cup in colour.

\*\*\*STEP SIX: Type in the AUCUP/LNW program and RUN it.

\*\*\*STEP SEVEN: Press Break and then type in the following line:

FLS:SAVE "AUCUP/GRF:d":

PLOAD "AUCUP/GRF:d"

where :d is drive number.

This will fill the screen enabling you to see the colour. The /GRF file will be saved and reloaded again so that you can verify that you have a good save.

For those who don't want colour, you will have to delete the following statements from the program:

Line 10 'OUT254,0:'

Line 140 'OUT254,116:'

Line 530 'OUT254,0:'

Line 630 'OUT254,0:'

A typical chain file to load in the colour would be as follows:

(Newdos80 2.0 users use

CHAINBLD/BAS)

LNWBASIC PLOAD "AUCUP/GRF"

CMD "S=BASIC"

RUN "AUCUP/BAS"

NOTE: AUCUP/BAS is the resulting pro-

gram when AUCUP/DAT and AUCUP/LST have been merged into one program.

### GRAFX L2/32K Disk

by Bob Wilson

#### WHAT IS GRAFX?

GRAFX is a tool for use in generating bold titles or graphics for incorporation in programs.

Once you have designed your titles or graphics, GRAFX will write a BASIC program to reproduce them.

#### INITIAL SET-UP

If you are typing this program in from the magazine, you must first type in the INIT program and run it. The INIT Program creates a disk file called BIGLTRS. This file gives GRAFX the ability to produce the big letters. GRAFX will require the BIGLTRS file to be on disk before it will run; once the INIT program has been run you will not need it again. If you are a disk subscriber you will not need to run INIT as the BIGLTRS file is already on your disk. The INIT program is also on the disk in case you should want to use it.

Make sure you have plenty of free disk space for the programs that GRAFX will write for you. Better still, prepare a disk containing only your operating system, GRAFX and BIGLTRS and initialize your AUTO command as BASIC RUN "GRAFX".

If you use NEWDOS 80 Ver 2 as your DOS you can run the program as it is. If you use another DOS, you will need to delete the last statement from line 80 of the program. The statement to delete is CMD "F";DELETE 10-70

Your Disk is now ready to run GRAFX on power-up or RESET.

#### USING GRAFX

Power-up or RESET with your GRAFX Disk in Drive 0.

GRAFX will grab as much of your memory as possible for use in string manipulation when it writes programs. It will do this calculation for itself. The program writing operation will take longer on a 32K '80 than on a larger machine, because the string area management functions (when string space becomes full) will operate more often.

Fifteen lines of your screen will be available for you to use in designing your titles or graphics. The bottom line is used for prompting you and to advise you of the current mode of operation.

After the initialization routine, the program will display the COMMAND MODE prompt line on the screen.

#### GRAFX COMMANDS

In COMMAND MODE the prompt line displayed is:

(C)lear (G)rafx (H)uge (S)lave (R)ecall (P)rogram (E)nd

The appropriate letter (C G H S R P or E) will select the various functions:

GRAFX mode

In GRAFX MODE you have several options available:

(C)ursor Mode

: Use the arrow keys to move the cursor

(D)raw Mode

: Use arrows to draw lines

(E)rase Mode

: Use arrows to erase lines

(T)ext Mode

: Enables text to be entered starting at the current cursor position

e(X)it

: Returns you to COMMAND MODE.

#### HUGE LETTERS MODE:

Enables HUGE letters to be entered for titles.

Screen capacity is 80 characters and the Character Set available is:

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
123456789!?:;.,-

#### NOTES

Input always starts at the top left of the screen.

If you reach the bottom of the screen, the top line will be scrolled off the screen and lost.

The screen will be cleared on entering this mode. Enter HUGE LETTERS BEFORE entering GRAPHICS.

To exit this mode enter SHIFT/BACKSPACE.

SAVE and RECALL MODES:

On selecting either of these modes you will be asked for a screen number.

The screen contents (15 lines) will be SAVED to, or RECALLED from, the screen memory you select and you will be returned to command mode.

#### PROGRAM MODE

You will be asked for one letter to be included in the FILESPEC. Your letter will appear as the 'x' in the following filespec:

TITLEx/GFX

A BASIC program will be written to reproduce the current screen constants, and will be dumped to disk in ASCII format.

The program created will contain 4 lines which will create and print a series of strings from data contained in the subsequent lines.

The END command terminates the running of GRAFX.

#### SOURCE UTILITY Model 3 Disk

by T. Domigan

SOURCE/BAS is a program to transfer EDTASM source files between tape and disk for Model 3 users of NEWDOS80 V2.0.

APPARAT discontinued support for Tape I/O in EDTASM for the Model 3 as they were concerned with the unreliability of tape. Whilst object code can be moved between tape and disk with LMOFFSET, source files cannot. SOURCE/BAS fills this gap and also allows the copying of source tapes.

To use SOURCE/BAS enter the command "BASIC IV" from DOS. This special command is necessary as a non-standard logical record length is used to get the code from disk. Once you have entered BASIC, "RUN" the program.

The machine language section resides in the bootstrap stack area and therefore does not require memory size to be set. However, memory is protected by the program in line 20, from 8000H to F000H for the storage of up to 26K of text.

This program is fully-menu driven and is self-explanatory.

Tape I/O can be made at high or low speed as desired. Disk I/O will tend to be slow so have patience with large files.

Always exit the program via menu option 6 as this ensures that the disk file is closed and BASIC is correctly restored.



## LVAR L2/32K

by Tim Fish

This is an extremely useful debugging tool for BASIC programs. The program resides at the top of memory but, on first being run, it initialises the DOS exit for the Disk BASIC verb NAME to a jump instruction to its own start address. Type NAME from ROM BASIC now, instead of giving you an L3 ERROR, gives you a list of all the single precision, double precision, integer and string variables used in your program and their values. So when your BASIC program fouls up press BREAK (if necessary) and type NAME. The variables are displayed in "pages" of 15. Press any key for the next page. Array variables are not listed.

### LOADING INSTRUCTIONS:

MEMORY SIZE 32320

SYSTEM \*? LVAR

\*? /ENTER or /32321

> - load your BASIC program (tape contains a daft demo program)

RUN ENTER ...  
... (BREAK) NAME ENTER

## TRACK RACER Hitachi Peach

by D.C. Kelly

In this race game you must steer your racing car down the track while avoiding the oncoming cars. You will crash if you run off the track or hit an oncoming car. The further you travel the greater your score. You steer the car using the left and right arrow keys. After you have crashed the program will display:

CRASH                      INS = PLAY AGAIN  
DEL = FINISH

If you want another game press the INS key, otherwise press the DEL key.

## HI-RES TEXT Colour Computer

by Geoffrey D. Williamson

One of the most annoying features of the CoCo is its lack of a true lower case display. It does not take too many hours in front of the monitor to become thoroughly annoyed with Tandy's excuse for lower case!

Unfortunately, this is fixed by hardware, and one may resort to alternative hardware lower case drivers to overcome this problem. Alas, these are expensive; yet there is a cheap way out of this dilemma, thanks to MICRO-80; Just type in the following program and you will be able to have true upper and lower case on the screen at the same time. Not only that, but this text is usable at the same time as graphics — here come some graphic adventures!!

One final feature of this program is its relocatability — by this, we mean it can live anywhere in memory quite happily, as it is written in position-independent code which is much easier to do on the 6809 than most 8 bit microprocessors.

Because of this relocatability you can 'hide' the machine code program behind the BASIC program — this is achieved by a little driver attached to the program proper. The advantage of all this is that once the whole program is set up,

it can ALL be loaded with one (C)LOAD, as if the machine code were not there!

This program is fully compatible with disk systems and 16K or 32K of memory. With it, you can use the PRINT@ command as you would on the text screen normally if the graphics screen is PMODE 4.

You can also use it with any start page, and there are some interesting effects in the coarser PMODE's — how about video titles using PMODE0 or 2?

However, the PRINT@ statements will not work normally except in PMODE 4. If you crash out of the program abnormally, and are still on the graphics screen, just type A=USR1(0) to get back to the text screen.

Type in the demo BASIC program — it is certainly simple, but does give you an idea of what is possible.

### PROGRAM DESCRIPTION

As the program is written in machine code it is very fast, and you would be hard up to find any slow down from that of the BASIC interpreter acting normally.

The source code is well commented and the remarks here are only intended for amplification.

The source code from line 120 to line 270 is all for relocating the machine code invisibly behind the BASIC program — invisible to the BASIC interpreter that is. Once that relocation is done the loader is dispensed with — in fact, it cannot be re-used — see line 250.

Lines 330 to 550 are for keeping tabs on the USR calls from the BASIC program. There is provision for two USR calls. The first (USR0) turns on the graphics display, and the second (USR1) turns it off. The code in this section is there to make sure that all errors from errant BASIC programmers are properly trapped!

The main code starts at line 610, and is commented extensively in the listing. You will note that no control characters are allowed in the listing, but these could be easily added if you wished to work out their parameters for the look up table.

### SET UP

1. Take very careful note of the addresses in the object code in the first twenty or so lines of code. If you are using Tandy's EDTASM+ to assemble this program you MUST force the assembler to use the direct page mode of addressing by using the < symbol. If you do not, the assembler will default to extended page addressing. This will slightly slow the code down, but more importantly, the addresses I have given in the demonstration program for the USR calls (and for the loader program) will be incorrect.

There is nothing sacrosanct about the values in the program, but be prepared to spend a fair bit of time in debugging if you do not stick slavishly to them!

2. Once you have the machine code safely assembled (and backed up) CLEAR some space at the top of RAM — if you have 16K try CLEAR200,&H3000: or CLEAR200,&H7000 for 32K. (C)LOADM the machine code into this area with the APPROPRIATE OFFSET. Remember, it is ORG'd at zero. If you forget the offset, you will bring down your whole system!

3. Now that the machine code is safely tucked away in protected RAM you can load in your BASIC program. Once you are happy with the latter you can go to the next step.

4. EXEC the machine code program. This places it behind the BASIC program and the WHOLE program can now be (C)SAVE'd like a normal BASIC Program.

5. RUN your new creation — I hope it works!!

## THE KILLER SATELLITE (COLOUR)

by Scott Edwards

"The Killer Satellite" is an original action game for the 16K, extended basic, colour computer. It should run on all colour computers with extended colour basic.

A war satellite in space has malfunctioned and is sending its deadly cosmic rays at Earth. You must protect Earth by destroying these rays until the satellite runs out of energy.

The cosmic rays are represented by a random sequence of coloured squares, all rapidly approaching your ship. The "colour" of your ship can be varied by pressing the space bar. By matching the colour of your ship to the components of the beam, portions of the beam (or ray) can be destroyed. If you destroy the whole ray, you go to the next level where a new ray moves faster. If the deadly beam reaches your ship, you are destroyed.

To begin the game you have 3 ships, and extra ships are obtained as you progress through the various levels. If all of your ships are destroyed before the satellite fails, the satellite destroys the Earth and you lose. On the other hand if you can protect the Earth for 11 rounds, the satellite runs out of power and the Earth is saved.

All this makes for an entertaining game which becomes quite difficult at the higher levels. For added interest a score is displayed at all times. If a ship is destroyed you continue on the same level until you are successful or otherwise.

To use the program simply type RUN, read the introduction, then hold down any key until the game begins. From then on simply use the spacebar to match the colours — Good Luck.

The functioning of the program is explained below:

Line 60, 70:

Generate initial display

Line 90, 100:

Generate the strings for the number of ships left, and the colour sequence of your ship.

Lines 110, 120:

Complete initial display

Lines 130-150:

Print scenario

Lines 160, 170, 180:

Flash screen, wait for continue

Line 200:

Initializes variables

Line 210:

Go to round generation

Lines 240-290:

Loops for colour mate between ship and ray by using INSTR ( ), then uses the MID\$ ( ) functions available in BASIC to manipulate the ray string.

Lines 310, 320:

Update display, and makes sound effects.

Line 340:

If 1st character of A\$ = blank, ray is destroyed, you win this round.

Line 370:

Update variables, if ray at your ship you lose round.

Line 390:  
Go to beginning of loop — continue round.

Lines 420-480:  
Check for spacebar being pressed, and change ship colour, if necessary. Also controls speed of ray.

Line 500:  
New round, if you complete 11 rounds then you've won, otherwise generate stars.

Line 510:  
If 5 successful rounds, get extra ship.

Line 520:  
Print satellite

Line 530:  
Print score, print round, generate ray.

Line 540:  
Time delay and colour sequence.

Line 560:  
Print remaining ships.

Line 580:  
Destroy ship, same round again.

Line 590:  
If last ship you lose

Lines 600-680:  
You lost, wait for key.

Line 690, 700:  
Another Game?

Line 720:  
You Won.

#### Variable List:

A\$ Cosmic Ray  
B\$ Colour sequence of ship  
Q\$ Displayed portion of A\$ (usually)  
X Position of left portion of Q\$ on screen  
SH Number of ships  
R Number of rounds  
M Score  
G, SV, S  
S1 Determines speed of ray  
Used to determine when to get new ship  
E Length of Q\$  
C Determines colour of ship  
Y1, X1 Position of starts  
Z, T, ZZ, V  
Variables used for — next loops.  
N.B. The use of INSTR\$( ) and MID\$( ) = . . . , both functions of extended BASIC allow this program to run at a reasonable speed. Therefore this program would be difficult to modify for computers without these special string handling routines.

all available drives for the file starting with drive 0.

In the case of a cassette load the program will read in the name of the first file found on the tape, if this does not match the filename you supplied the program will display the filename read from tape and abort the function.

After a successful load from either disk or cassette the program will return to the LOAD MENU.

### 3. READ MEMORY

This function scrolls ALL data stored in memory, up the screen.

### 4. EDIT MEMORY

This function is used to add new records, or to change the data stored in any one record. It uses the same input and display format as the keyboard input mode except that it gives several extra options:

Respond to the prompt SELECT with one of the following:

- E This will put you into edit mode. The input format is the same as the Keyboard input function EXCEPT that this time if the RETURN key is pressed at the start of an entry field, that field will remain unchanged and input will move to the next field.
- ; This will increase the record count by one and display the next record.
- + This will increase the record count by ten and display a higher record.
- This will decrease the record count by one and display the preceding record.
- = This will decrease the record count by ten and display a lower record.

RETURN in response to the SELECT \* prompt will return you to the main menu.

### 5. SAVE DATA

This function enables you to save a complete set of records. The program will display a menu giving you the choice of saving to cassette or disk, or of returning to the main menu should you have selected this function by mistake. For either type of save the program will request an 8 character filename, write the file and return to the SAVE MENU. (You should make a note of the filename you use, particularly when saving to cassette, as the program requires the name when you wish to reload the data).

### 6. PRINT JOURNALS

This function enables you to print the four journals to the screen and to a printer (optional). Follow the instructions on the screen. Note: Do not attempt to use the printer option unless there is a printer connected to the computer. Pressing RETURN in response to the DATE prompt will cause all dates for that journal to be printed.

### IMPORTANT NOTE

Before the Ledger Balances are calculated and printed, the records in memory are sorted according to Account No. This destroys the normal DATE-order sequence of the record in memory, which is the preferred order for all other printouts. **Make sure that you SAVE a copy of your data before using this function.**

### 7. LINEPRINTER UTILITY

This function allows direct interface between the keyboard and the printer, i.e. whatever is typed onto the screen will be printed as soon as the RETURN key is pressed. This utility is useful for setting up headings on reports. Note: Do not attempt to use this function unless there is a printer connected to the computer.

## HOUSEHOLD ACCOUNTING VER 4.2 & VER 5.0

This is an all new version of the earlier Household Accounts program published in MICRO-80 magazine. Input has been simplified, all bugs eliminated and, the program will load and save to disk or tape as required. The program will carry out many of the bookkeeping functions for a small business.

### LOADING FROM CASSETTE

The program is written in BASIC. Simply position the tape to the start of the

program, type LOAD, press RETURN and press the PLAY key on the recorder. The program will then load. When loaded, type RUN and press RETURN. The Household Accounting copyright message will be displayed together with the program menu.

### FUNCTIONS

The following MENU is displayed when the program starts:

```

* * * H O U S E H O L D   A C C O U N T I N G   V E R 4.0 * *
* * * (C) 07/07/83      M I C R O - 8 0   P T Y L T D * *

```

#### MENU

```

1 = KEYBOARD INPUT      : 5 = SAVE DATA
2 = LOAD DATA          : 6 = PRINT JOURNALS
3 = READ MEMORY         : 7 = LINEPRINTER UTILITY
4 = EDIT MEMORY         : 8 = LEDGER ACCOUNTS

```

### 1. KEYBOARD INPUT

This function is for the input of data and starts from record 1 and works up to the maximum record number for your system. Re-entry to this function when data is still resident in memory will cause the program to resume at the next record after your last entry (this also applies where a file is loaded from disk or tape). Your input is displayed next to the request for details. The cursor is displayed at the point where your entry is going to take place. The asterisks in each input field indicate the maximum number of characters that the field will accept. When a field has been filled, your input will move to the next field. You do not have to fill each field. When you have entered the information you require simply press RETURN and the program will move on to the next field. The only exception to this is the first input of DATE \*\*/\*\*/\*\*. If RETURN is pressed in response to the first input field the program will return you to the menu.

The field "PREFIX" is used to indicate to the program the ledger into which the record is to be placed. This should be one of the following:

CP = Cash Payments Journal  
CR = Cash Received Journal

GJ = General Journal  
SJ = Sales Journal

Input to the numeric field of DEBIT and CREDIT must have leading zeroes, or blanks.

i.e. enter:

\$10.10 as 00010.10  
\$10.01 as 00010.01  
\$1.10 as 00001.10  
\$101.10 as 00101.10

When all fields have been filled the program will respond with:  
CORRECT (Y/N)

If 'N', the program will return you to the start of input for that record. If 'Y', the program, will respond by going to the next record. To return to the menu press RETURN in response to the DATE field input.

### 2. LOAD DATA

This function enables you to load a complete set of data. The program will display a menu giving you the choice of loading from cassette or disk, or of returning to the main menu should you have selected this function by mistake.

For either type of load the program will request an 8 character filename. In the case of a disk load the program will search



**8. PRINT LEDGER ACCOUNTS**

This function allows you to print the ledger accounts to the screen and to a printer (optional). Follow the instructions on the screen. Note: Do not attempt to use the printer option unless there is a printer connected to the computer.

**USING THE PROGRAM**

The program has the capacity to handle both single entry and double entry accounting systems. The benefit of the double entry system is that it provides a checking mechanism to indicate accuracy of input and processing and is the preferred system in maintaining financial records. However, the double entry system uses more records in recording the data thereby reducing the total capacity available for recording transactions.

For the purposes of this illustration the double entry system has been used.

Prior to using the program, it is necessary to establish a Chart of Accounts to accumulate transactions appropriate for the purpose of recording. Adequate allowance has been made for the number of ledger accounts (998) that may be used. The following example has been used to illustrate how the program can be used to record personal income and expenditure.

**CHART OF ACCOUNTS****Balance Sheet of Items****Assets**

1. Land
2. Buildings
3. Household Furniture & Appliances
4. Motor Vehicles

6. Public Co. Shares
7. Bank Accounts

**Liabilities**

11. 1st Mortgage
12. 2nd Mortgage
14. Credit Charge Accounts
15. Friendly Finance Corp
16. Other Liabilities

99. Owner's Equity.

**Income and Expenditure Items****Income**

51. Husband's Salary
52. Wife's Salary
54. Dividends Received
55. Interest Received
58. Health Refunds
59. Other Income

**Expenditures****General**

71. Entertainment
72. Household Repairs
73. Housekeeping
74. Insurance
75. Motor Vehicle Costs
76. Subscriptions
77. Interest paid
78. Health Insurance

**Taxation Deductions**

81. Education
82. Medical Costs
83. Rates & Taxes
84. Trade Subscriptions

To commence processing the transactions shown in the example the opening balances need to be entered. This is done via the keyboard input. For this purpose the General Journal (GJ) prefix is used.

(N.B. The print out of the General Journal processing demonstrates the self checking mechanism of the double entry

system in showing that the total of debits and credits agree and that there is no imbalance. If a figure other than \$0.00 is

shown as the balance this indicates an error in processing has occurred and requires investigation and editing).

**CASH RECEIPTS JOURNAL****CASH RECEIVED JOURNAL FOR ALL (DATE)'S**

DATE	REF	DETAILS	ACC NO	DEBIT	CREDIT
15/07/83		MR. SALARY	CR051	0.00	1050.00
31/07/83		MRS. INCOME	CR052	0.00	600.00
01/08/83		DIV. A CO. LTD.	CR054	0.00	150.00
15/08/83		MR. SALARY	CR051	0.00	1050.00
17/08/83		HEALTH REBATE	CR058	0.00	10.60
27/08/83		BEQUEST NELLY	CR059	0.00	1000.00
31/08/83		INT. X CO. LTD.	CR055	0.00	450.00
31/08/83		MRS. INCOME	CR052	0.00	600.00
31/08/83		TOTAL RECEIPTS	CR010	13910.60	0.00
TOTAL				13910.60	13910.60
BALANCE				\$0.00-	

Normally for each receipt an entry is required to the Bank Account (debit entry) and the ledger account to record the nature of the income (credit entry). In this example only one entry has been made to the Bank Account for the total of the records processed. This has been done to demonstrate a way in which maximum availability of records can be maintained. To carry out this function return to the menu and select the "Print Journals"

function and list the transactions to obtain the total of the input. Once obtained, return to the "Keyboard Input" function and process this total to the Bank Account using the "CR" prefix.

The comments noted above for Cash Receipts also apply here, except that the entry to the Bank Account for cash payments is a credit and entries to the expense accounts, etc. a debit.

**CASH PAYMENTS JOURNAL****CASH PAYMENTS JOURNAL FOR ALL (DATE)'S**

DATE	REF	DETAILS	ACC NO	DEBIT	CREDIT
01/07/83	0001	CAR SERVICE	CP075	43.24	0.00
10/07/83	0002	ENT-BILL MARY	CP071	75.40	0.00
15/07/83	0003	MICRO-80	CP076	85.00	0.00
15/07/83	0004	HOUSEKEEPING	CP073	400.00	0.00
31/07/83	0005	BUILDING INS.	CP074	230.63	0.00
31/07/83	0005	CONTENTS INS.	CP074	250.00	0.00
31/07/83	0006	A BANK - PRIN.	CP011	80.00	0.00
31/07/83	0006	A BANK - INT.	CP077	718.75	0.00
31/07/83	0007	A SBANK - PRIN.	CP012	100.00	0.00
31/07/83	0007	A SBANK -INT.	CP077	533.00	0.00
31/07/83	0008	FRIENDLY FINAN.	CP015	1000.00	0.00
01/08/83	0009	DR. WHO	CP082	10.60	0.00
01/08/83	0010	FRIENDLY FINAN.	CP015	9000.00	0.00
01/08/83	0011	A SBANK - PRIN.	CP012	1000.00	0.00
10/08/83	0012	SCH. FEES - ANN	CP081	100.00	0.00
15/08/83	0013	COUNCIL RATES	CP083	212.10	0.00
15/08/83	0014	HOUSEKEEPING	CP073	400.00	0.00
31/08/83		TOTAL PAYMENTS	CP010	0.00	14238.72
TOTAL				14238.72	14238.72
BALANCE				\$0.00	

**LEDGER BALANCES (TRIAL BALANCE)**

ACC NO.	DEBITS	CREDITS	TOTAL
001	10000.00	0.00	10000.00
002	65000.00	0.00	65000.00
003	25000.00	0.00	25000.00
004	19000.00	0.00	19000.00
006	5000.00	0.00	5000.00
007	15000.00	0.00	15000.00
010	14135.76	14238.72	102.96-
011	80.00	25000.00	24920.00-
012	1100.00	40000.00	38900.00-
015	10000.00	10000.00	0.00
051	0.00	2100.00	2100.00-
052	0.00	1200.00	1200.00-
054	0.00	150.00	150.00-
055	0.00	450.00	450.00-
058	0.00	10.60	10.60-

059	0.00	10000.00	10000.00-
071	75.40	0.00	75.40
073	800.00	0.00	800.00
074	480.63	0.00	480.63
075	43.24	0.00	43.24
076	85.00	0.00	85.00
077	1251.75	0.00	1251.75
081	100.00	0.00	100.00
082	10.60	0.00	10.60
083	212.10	0.00	212.10
099	0.00	64225.16	64225.16-
<b>TOTAL</b>	<b>167374.48</b>	<b>167374.48</b>	<b>0.00</b>

This facility summarises the balances of all active ledger accounts showing the total debit and credit entries and the balance of each account. Validation of the processing is ascertained when the total debits and credits are the same amount and the total equals 0.00. If the total does not equal 0.00 an imbalance has occurred which requires checking and correction.

Normally, each of the four journals would be individually checked prior to performing a trial balance to ensure that each journal is itself balanced. The trial balance is then merely a formality and should show no imbalance.

#### IMPORTANT NOTE

Before the Ledger Balances are calculated and printed, the records in memory are sorted according to Account No. This destroys the normal DATE-order sequence of the records in memory, which is the preferred order for all other printouts. **Make sure that you SAVE a copy of your data before using this function.**

#### LEDGER ACCOUNTS

This function enables each individual account to be printed showing all the transactions processed to that account. For the purposes of illustration the Bank Account (#10) and the Interest Paid Account (#77) are listed below.

To obtain a complete print out of ledger accounts it is necessary to input each account number in response to the "Which Account Number do you require" question.

In cases where the record maximum is reached this can be overcome by obtaining a print out of all journals and ledger accounts for retention as a permanent record and repeating the procedural functions outlined above.

This commences by clearing the memory and inserting the closing balances for each ledger account via the General Journal prefix.

The program provides great flexibility as to the types of accounts that may be incorporated into the chart of accounts. Your requirements can be specifically designed for your own particular circumstances. The design and allocation of account numbers should be thoughtfully considered prior to commencement as any subsequent change in account numbers for particular purposes will require the editing of prior transactions to accounts affected.

#### SALES JOURNAL

The Sales Journal will often not be applicable in the normal household situation. It is, of course, useful for a small business.

Version 4.2 has been modified to run on the Hitachi Peach.

Version 5.0 has been modified to run on the TRS-80 Model 4.

Instructions for the Model 4 version are the same as those for the Hitachi Peach, the program has however been fur-

ther modified to make use of the Model 4 features: the description field has been expanded to 31 characters and the program has been broken up into modules to gain maximum use of the memory. Model 4 owners should type in each module from the magazine and save it to disk as an ASCII file using the filename that is listed in the module, ie: SAVE "MODULE0/BAS",A

When the program is run, MODULE0 must be the first program that is loaded and run, this module initializes the system. The program will load the other modules as required.

Model 4 users should ignore any references in the main text to cassette I/O, these functions are not supported by your machine. Model 4 owners MUST be in the Model 4 mode to run this program.

#### BOLD PRINTING ON LINE PRINTER VII

This 1 line subroutine which enables owners of a Lineprinter VII to use bold type. The subroutine may also work on other Printers, but I have not tried them.

To use the subroutine, simply LPRINT your normal text, and when you want to use bold type, put the text in P\$ and use GOSUB 1000 (or wherever you put the subroutine). The text must not go over on to the next line, or an error will result. On returning from the subroutine, the Printer's carriage will be placed just after the last character of P\$. The routine will work in normal and double size Print mode, but if both are mixed on one line, problems will result.

The subroutine works by simply printing P\$ in the same place three times. The control code 26 executes a carriage return with no line feed. Z\$\$ is a string of blanks, which will return the carriage to the start of P\$ after each carriage return. ZP is used to check that P\$ will fit on one line and to calculate the length of Z\$\$\$. If you want the bold type darker, increase the length of the FOR ... NEXT loop.

#### SIRIUS ADVENTURE for the Colour Computer and Hitachi Peach

This Adventure was originally written to run on the Model 1.

It has mainly been converted to take advantage of the HI-RES WRITER published elsewhere in this issue. If you are typing this program in from the magazine, follow the instructions for merging the HI-RES WRITER with this program and then save it. If you have a cassette subscription you only to load the program and the HI-RES WRITER will load into memory as well.

The adventure is fairly small by normal standards, though it will still provide you with a challenge. The Hitachi Peach version is essentially the same program, but without the need for the HI-RES WRITER. As is usual with Adventures, no further clues to its operation will be given here, you'll just have to sweat it out!!!

#### SPACE UTILITY DISC BASIC by D. Bereis

This utility will go through a Basic program and insert spaces around all of the keywords, this not only makes the listing easier to read on a Model 1, but if you have purchased a Model 4 and still have your Model 1 it will be essential if you want to convert your old Model 1 programs, since the Model 4 WILL NOT run a program that does not have spaces between the keywords. Using this utility will save you hours of typing. To use it simply load your Basic program then type: CMD"S"

Then type:

SPACE and press ENTER/NEWLINE  
Then type:

BASIC \* and press ENTER/NEWLINE

Then save the program. If you have a Model 4, all you have to do now is put the disk in your Model 4 and run the conversion program on it. Remember you will still have to check for things like PEEK's and POKE's before you run it.

#### YAHTZEE L2/16K by T. Domigan

YAHTZEE is based on the popular poker-like 5-dice game of the same name.

#### AIM

The aim of the game is to score in each of 13 categories and the winner is the player with the highest total score.

#### ROLLING THE DICE

Each player is allowed 3 rolls of the dice to maximise the score in each of the 13 turns. The first roll changes all 5 dice whilst the remaining 2 players allow selective rolling of any of the 5 dice, e.g. if Rolls 1 gives 4, 3, 6, 1, 4, to maximise the score by rolling more 4's, then "R234" would be suitable. If a good score results after the first or second roll then you may immediately score by entering "S" instead of "R". When scoring it is only necessary to enter the category number e.g. 13.

#### SCORING

Categories 1-6: Sum of that category dice in hand.

Category 7: 3 of a kind. Score total of dice.

- 8: 4 . . . . . 25 pts
- 9: Full house. . . 25 pts
- 10: Small Straight (sequence of 4 dice) 30 pts
- 11: Large Straight (sequence of 5 dice) 40 pts
- 12: YAHTZEE (5 of a kind) score 50 pts
- 13: CHANCE = anything

score sum of dice  
If a second or later YAHTZEE (5 of a kind) is scored and the appropriate category 1-6 has been filled, it may be used in categories 7-11.

Sound is available but memory need not be protected as the m/l will reside in the REM statement of line 10.

NOTE: Although written for a Model 3, this program will run on a Model 1 without modification.

#### SPACE INVADERS for the TRS-80 MC-10

This short Space Invaders game is included in case any of you have bought an MC-10 for Christmas, the program is short and simple, (out of necessity due to the limited memory). To move your laser cannon left and right press the "A" or the "S" keys respectively, these keys were chosen because they have the left and right arrow symbols on them as well. Press the Space Bar to fire. Ten points are given for each Invader that is destroyed. If you shoot all the Invaders before they land you get a bonus of 30 points. If any Invaders land, you lose.



```

550 L=1:FORI=1TOW-1
560 GOSUB1570:GOSUB1620:PRINTVX$
570 L=L+1:IFL=11THENL=1:LOCATE0,15:PRINT
580 PA=986:LN=1:GOSUB60:60%:3230:GOSUB5
10
580 NEXTI
590 LOCATE0,15:PRINT"END OF DATA - "S0$
:PA=1000:LN=1:GOSUB60:RETURN
600 I=1:FL=0
610 LOCATE28,3:PRINT"EDIT "I21$
620 GOSUB230:GOSUB310:LOCATE0,13:PRINT"
":LOCATE0,15:PRINT"SELECT
**I
630 LOCATE0,4:PRINT"RECORD NO. "I1$
640 GOSUB1570
650 IFV1$=""THENGOSUB310:LOCATE0,13:PRIN
T"
":
660 LOCATE8,5:PRINTV1$:LOCATE11,5:PRINT
V2$:LOCATE14,5:PRINTV3$:LOCATE8,6:PRIN
TV4$:LOCATE8,7:PRINTV5$:LOCATE8,8:PRIN
TV6$:LOCATE8,9:PRINTV7$:LOCATE7,10:PRI
NTV8$:LOCATE7,11:PRINTV9$
670 PA=967:LN=1:GOSUB60:IFAD$=""THENLOC
ATE0,15:PRINT"ADVANCE":I=I+1:IF I>VV TH
EN I=VV
680 IF AD$=""THENLOCATE0,15:PRINT"ADVAN
CE":I=I+1:IF I>VV THEN I=VV
690 IF AD$=""THEN I=I-1:LOCATE0,15:PRIN
T"REVERSE":IF I<0 THEN I=1
700 IF AD$=""THEN I=I-10:LOCATE0,15:PRI
NT"REVERSE":IF I<0 THEN I=1
710 IFAD$=""THEN250
720 IF AD$=""THENLOCATE0,15:PRINT"* EDI
T *":GOSUB840:GOTO620
730 GOTO630
740 GOSUB230:LOCATE18,3:PRINTZ4$:"S AVAI
LABLE"
745 LOCATE0,5:PRINT"1 = PRINT LEDGER BAL
ANCES : 4 = GJ GENERAL "I24$:LOC
ATE0,6:PRINT"2 = CP CASH PAYMENTS "I24$
": 3 = SJ SALES "I24$:LOCATE0,7:PR
INT"3 = CR CASH RECEIVED "I24$": 6
= RETURN TO MAIN MENU"
750 LOCATE0,13:PRINTZ3$:PA=848:LN=1:GOS
UB60:AD=VAL(AD$)
760 IFAD<1 OR AD>6 THEN750
770 ON AD GOTO 1630,780,790,800,810,250
780 PT$="CASH PAYMENTS":KAS="CP":GOTO820
790 PT$="CASH RECEIVED":KAS="CR":GOTO820
800 PT$="GENERAL":KAS="GJ":GOTO820
810 PT$="SALES":KAS="SJ"
820 GOSUB980:GOSUB1010:GOTO1030
830 PRINTS0$:"A=1022:LN=1:GOSUB60:GOTO25
0
840 FL=1:PA=328:LN=2:GOSUB60:IFAD$=""T
HENDT$=V1$+V2$+V3$:LOCATE8,5:PRINTV1$:"/
":V2$:"/":V3$:GOTO860
850 DT$=AD$:GOSUB370
860 GOSUB380:IFR$=""
":THENR$=V4$:LOC
ATE8,6:PRINTR$

```

```

250 P=0:GOSUB230:LOCATE28,3:PRINT"MENU"
255 LOCATE1,6:PRINT"1 = KEYBOARD INPUT
: 5 = SAVE DATA":LOCATE1,7:PRIN
T"2 = LOAD DATA : 6 = PRI
NT JOURNALS"
260 LOCATE1,8:PRINT"3 = READ MEMORY
: 7 = LINE PRINTER UTILITY":LOC
ATE1,9:PRINT"4 = EDIT MEMORY
: 8 = PRINT LEDGER "I22$:"5"
270 LOCATE0,13:PRINTZ3$:PA=848:LN=1:GOS
UB60
280 AD=VAL(AD$):IFAD<1 OR AD>8 THEN270
290 ON AD GOTO480,1450,500,600,1350,740,
1150,1180
300 LOCATE18,3:PRINT"KEYBOARD INPUT"
310 LOCATE0,5:PRINTZ7$:" **/**/**:L
OCATE0,6:PRINT"REF NO. ***":LOCATE0,7:
PRINT"DETAILS *****":LOCATE0,
8:PRINT"PREFIX **":LOCATE0,9:PRINT"ACC
NO. ***":LOCATE0,10:PRINTZ6$:" ****.
**"
320 LOCATE0,4:PRINT"RECORD NO. "I1$
330 LOCATE0,11:PRINTZ5$:" ***.***:LOC
ATE0,13:PRINT"CORRECT (Y/N) *":RETURN
340 GOSUB230:GOSUB300
350 FL=1:PA=328:LN=2:GOSUB60:DT$=AD$:IFD
T$=""THENRETURN
360 GOSUB370:GOSUB380:GOSUB390:GOSUB400:
GOSUB410:FL=1:GOSUB420:GOSUB430:GOSUB440
:GOSUB450:FL=0:GOTO460
370 PA=331:GOSUB60:DT$=DT$+AD$:PA=334:GO
SUB60:DT$=DT$+AD$:RETURN
380 PA=392:LN=4:FL=0:GOSUB60:RF$=AD$:RET
URN
390 PA=456:LN=15:GOSUB60:DE$=AD$:RETURN
400 PA=520:LN=2:GOSUB60:PR$=AD$:RETURN
410 PA=584:LN=3:GOSUB60:PR$=PR$+AD$:RETU
RN
420 PA=647:LN=5:GOSUB60:GOSUB190:DB$=AD$
+":"RETURN
430 PA=653:LN=2:GOSUB60:DB$=DB$+AD$:RETU
RN
440 PA=711:LN=5:GOSUB60:GOSUB190:CR$=AD$
+":"RETURN
450 PA=717:LN=2:GOSUB60:CR$=CR$+AD$:FL=0
:RETURN
460 PA=846:LN=1:GOSUB60:IFAD$=""THEN340
ELSEIFAD$<"Y"THEN460
470 A$(1)=DT$+RF$+DE$+PR$+DB$+CR$:RETURN
480 FORI=1TOW VV:GOSUB340:IFDT$=""THENW
=1:GOTO250
490 NEXTI:GOTO250
500 GOSUB230:GOSUB510:GOSUB550:GOTO250
510 LOCATE25,3:PRINT"CONTENTS OF "I21$
520 LOCATE0,4:PRINTZ7$:" REF DETAI
LS ACC NO "I26$:" "I25$
530 IF P THEN PRINT#3,Z7$:" REF DE
TAILS ACC NO "I26$:" "I
25$
540 RETURN

```

\*\*\* VER 4.2 HOUSEHOLD ACCOUNTING \*\*\*

HITACHI PEACH

```

10 REM WRITTEN BY
SUNBURST SOFTWARE SERVICES
FOR
MICRO-80 PTY LTD
20 CLEAR 16100:SCREEN0,1:VV=350:POKE&H2
3C,16:POKE164,10
30 DIM A$(351),LB(351):F3$=""
: F4$="" $$$$,$$,$$-:F5$=""
: F4$="" $$$$,$$,$$-:F5$=""
40 W=1:S0$="PRESS ANY KEY TO CONTINUE "
:Z0$="TOTAL":Z1$="MEMORY":Z2$="ACCOUNT":
Z3$="SELECT FUNCTION *":Z4$="JOURNAL":Z5
$="CREDIT":Z6$="DEBIT":Z7$="DATE"
50 GOSUB230:LOCATE10,10:PRINT"MAXIMUM NU
MBER OF RECORDS = "VV:LOCATE10,13:PRINT
S0$:PA=848:LN=1:GOSUB60:GOTO250
60 AD$=""W=INT(PA/64):WY=PA-(WX*64)
70 FORT=1TOLN
80 GOSUB150:IFIN$=CHR$(13)THEN130ELSEIFI
N$=CHR$(8)THEN110ELSEIFIN$=CHR$(32)THENG
OSUB170
90 AD$=AD$+IN$:LOCATE WY,WX:PRINTAD$:NE
XT:RETURN
100 NEXT:RETURN
110 IF<=1THEN80ELSESET=T-1
120 AD$=LEFT$(AD$,LEN(AD$)-1):LOCATE WY,
WX:PRINTAD$:"**":GOTO80
130 IF FL=0 THENBL$=STRING$(LN-LEN(AD$),
" "):AD$=AD$+BL$:LOCATE WY,WX:PRINTAD$:
RETURN
140 BL$=STRING$(LN-LEN(AD$),"0"):AD$=AD$
+BL$:LOCATE WY,WX:PRINTAD$:RETURN
150 IN$="":IN$=INKEY$:GOSUB160:IFIN$=""T
HEN150ELSERETURN
160 LOCATE WY,WX:PRINTAD$:CHR$(13):RE
TURN
170 IFFL=0THENRETURN
180 IN$="":RETURN
190 T=1
200 IF<=5THENAD$=""
210 IFMID$(AD$,T,1)=""THENT=T+1:GOTO200
220 AD$=STRING$(T-1,32)+RIGHT$(AD$,6-T):
RETURN
230 CLS:LOCATE0,0:PRINT"* * * H O U S
E H O L D A C C O U N T I N G V E R 4.2
* * *":LOCATE0,1:PRINT"* * * (C)
07/07/83 M I C R O - 8 0 P T Y L
T D * * *
240 LOCATE1,3:RETURN

```

```

1550 LOCATE0,13:PRINT"THIS IS FILE "I;N1$
I; NOT FILE "I;N2$;CLOSE#2
1560 LOCATE0,14:PRINT"PRESS ANY KEY TO A
BORT *I;PA=919:LN=1:GOSUB60:GOTO1450
1570 V1$=LEFT$(A$(I),2):V2$=MID$(A$(I),3
,2):V3$=MID$(A$(I),5,2):V4$=MID$(A$(I),7
,4):V5$=MID$(A$(I),11,15):V6$=MID$(A$(I)
,26,2):V7$=MID$(A$(I),28,3):V8$=MID$(A$(I)
,31,8):V9$=MID$(A$(I),39,8):RETURN
1580 GOSUB230:LOCATE23,3:PRINT"DATA "I$
FT$(X1$,4):LOCATE0,5:PRINT"1 = "I;X1$ "I
APE":LOCATE0,7:PRINT"2 = "I;X1$ "DISK":L
OCATE0,9:PRINT"3 = EXIT TO MENU":RETURN
1590 LOCATE0,13:PRINTZ3$:PA=848:LN=1:GO
SUB60:SF=VAL(AD$):IF SF<1 OR SF>3THEN159
0ELSERETURN
1600 FL=0:GOSUB230:LOCATE0,7:PRINT"ENTER
FILENAME *****I;PA=463:LN=8:GOSUB60
:NMS$="I;"+AD$:RETURN
1610 LOCATE0,8:PRINT"PRESS ANY KEY WHEN
DEVICE READY OR (E)SCAPE *I;PA=558:LN
=1:GOSUB60:RETURN
1620 V1$=V1$+"I;"+V2$+"I;"+V3$+"I;"+V4$+"
"+V5$+"I;"+V6$+"I;"+V7$+"I;"+V8$+"I;"+V9$:
RETURN
1630 GOSUB230:GOSUB980:GOSUB230:GOSUB178
0:LOCATE15,7:PRINT" * * * W A I T * * *
":FOR I=1 TO W-1:GOSUB1570:LB(I)=VAL(RIG
HT$(V7$,3)):NEXTI:J=0:CT#=#0:BL#=#0
1640 GOSUB230:PRINT"PRINT"ACC NO."I;TAB(1
6):Z6$;"S";TAB(34):Z5$;"S";TAB(55):Z0$
1650 IF P THEN PRINT#3," ":PRINT#3,"ACC
NO."I;TAB(16):Z6$;"S";TAB(34):Z5$;"S";TAB
(55):Z0$
1660 TT#=#0:DR#=#0:CR#=#0:J=J+1
1670 VF=LB(J):FORI=1TOW:IFLB(I)=VF AND L
B(I)<>THENGOSUB1770
1680 NEXTI:LB(J)=0:IFJ=WTNEN1740ELSE1690
1690 IFVF=0THEN1660
1700 TT#=#0:DR#=#0:CT#=#0:CT#=#0:CR#=#0
:BL#=#0:TT#
1710 PRINT" "I;MID$(A$(J),28,3):PRINTUS
INGF5$;DR#;CR#;TT#
1720 IF P THEN PRINT#3," "I;MID$(A$(J),2
8,3):PRINT#3,USINGF5$;DR#;CR#;TT#
1730 GOTO1660
1740 PRINT:PRINTZ0$;PRINTUSINGF5$;DT#;C
T#;BL#
1750 IF P THEN PRINT#3,:PRINT#3,Z0$;:PRI
NT#3,USINGF5$;DT#;CT#;BL#;CLOSE#3
1760 LOCATE0,15:PRINTS0$;PA=986:LN=1:GO
SUB60:GOTO250
1770 GOSUB1570:DR#=#0:VAL(V8$):CR#=#0:CR#
VAL(V9$):LB(I)=0:RETURN
1780 LOCATE15,7:PRINT" * * * SORTING *
* *":FOR SC=1 TO W-1:FOR SA=1 TO W-1
1790 SA$=MID$(A$(SA),28,3)
1800 SB$=MID$(A$(SA+1),28,3):IF SB$<SA$
ENGOTO1810ELSE IF SA$>SB$ THEN SB$=A$(SA
):A$(SA)=A$(SA+1):A$(SA+1)=SB$
1810 NEXTSA:NEXTSC:RETURN

```

```

1180 GOSUB230:LOCATE18,2:PRINT"LEDGER "I
Z2$;"S":LOCATE0,5:PRINT"TYPE "I;CHR$(34)
"999";CHR$(34); " TO EXIT"
1190 GOSUB980
1200 LOCATE0,15:PRINT"WHICH "I;Z2$;" NO.
DO YOU WANT (AD$)
SUB60:N=VAL(AD$)
1210 IFN<1ORN>999THEN1200
1220 BL#=#0:DT#=#0:CT#=#0:IFN=999THENCLOSE#
3:GOTO250
1230 GOSUB240:LOCATE0,3:PRINTZ2$;" NO. "
I;AD$:IF P THEN PRINT#3," ":PRINT#3,Z2$;
" NO. "I;AD$
1240 LOCATE0,6:GOSUB520:FORI=1TOW
1250 IFN<VAL(MID$(A$(I),28,3))THEN1280
1260 GOSUB1570:GOSUB1620:PRINTVX$;IF P T
HEN PRINT#3,VX$
1270 DR#=#0:VAL(MID$(A$(I),31,8)):CR#=#0:VAL(M
ID$(A$(I),39,8)):DT#=#0:DR#=#0:CT#=#0:CR#
:BL#=#0:DR#=#0:CR#
1280 NEXT
1290 PRINT:PRINTZ0$;TAB(36):PRINTUSINGF
3$;DT#;CT#
1300 IF P THEN PRINT#3," ":PRINT#3,Z0$;I
AB(36):PRINT#3,USINGF3$;DT#;CT#
1310 PRINTZ2$;" BALANCE" I;PRINTUSINGF4$;
BL#
1320 IF P THEN PRINT#3,Z2$;" BALANCE" I;P
RINT#3,USINGF4$;BL#
1330 PRINT:PRINT:IF P THEN PRINT#3," ":P
RINT#3," "
1340 GOTO1200
1350 X1$="SAVE TO":GOSUB1580
1360 GOSUB1590
1370 IF SF=3THEN250
1380 GOSUB1600
1390 GOSUB1610:IF AD$="E" THEN GOTO1350
1400 IF SF=1 THEN N1$=RIGHT$(NMS,8):OPEN
"O",#2,"CAS0:DATA":PRINT#2,N1$,W
1410 IF SF=2 THEN OPEN"O",1,NMS
1420 IF SF=1 THEN FOR I=1 TO W STEP4:PRI
NT#2,A$(I),A$(I+1),A$(I+2),A$(I+3):NEXT:
CLOSE#2
1430 IF SF=2 THEN PRINT#1,W:FOR I=1 TO W
:PRINT#1,A$(I):NEXT:CLOSE
1440 GOTO1350
1450 X1$="LOAD FROM":GOSUB1580
1460 GOSUB1590
1470 IF SF=3THEN250
1480 GOSUB1600
1490 GOSUB1610:IF AD$="E" THEN GOTO1450
1500 IF SF=1 THEN N2$=RIGHT$(NMS,8):OPEN
"I",#2,"CAS0:DATA":INPUT#2,N1$,W:IFN1$<
N2$THENGOTO1550
1510 IFSF=2THENOEN" I",1,NMS
1520 IF SF=1 THEN FOR I=1 TO W STEP4:INP
UT#2,A$(I),A$(I+1),A$(I+2),A$(I+3):NEXTI
:CLOSE#2
1530 IF SF=2 THEN INPUT#1,W:FOR I=1 TO W
:INPUT#1,A$(I):NEXT:CLOSE
1540 GOTO1450

```

```

870 GOSUB390:IFDE$="
DE$=V5$:LOCATE8,7:PRINTDE$;
880 GOSUB400:IFPR$=" " THENP1$=V6$:LOCAT
E8,8:PRINTV6$;GOTO990
890 P1$=AD$
900 GOSUB410:IFAD$=" " THENPR$=P1$+V7$:
LOCATE8,9:PRINTV7$;GOTO920
910 PR$=P1$+AD$
920 FL=0:GOSUB420:IFAD$=" " THENMD$=V
8$:LOCATE7,10:PRINTV8$;GOTO940
930 GOSUB190:GOSUB430
940 FL=0:GOSUB440:IFAD$=" " THENCR$=V
9$:LOCATE7,11:PRINTV9$;GOTO960
950 GOSUB190:GOSUB450
960 GOSUB470
970 FL=0:RETURN
980 LOCATE0,13:PRINT"IS THE PRINTER REQU
IRED (Y/N) *I;PA=862:LN=1:GOSUB60
990 IF AD$<"N" AND AD$<"Y" THEN980
1000 P=(AD$="Y"):IF P THEN OPEN"O",#3,"L
PT0:"
1005 RETURN
1010 ZK$="":LOCATE0,14:PRINT"WHICH "I;Z7$
I; DO YOU REQUIRE *I;*//*I;FL=1:PA=922
:LN=2:GOSUB60:E$=AD$:IF E$="0" THEN E$="S
":ZK$="ALL ":GOTO1030
1020 PA=925:GOSUB60:E$=E$+AD$:PA=928:GOS
UB60:E$=E$+AD$:RETURN
1030 GOSUB230:LOCATE0,3:PRINTPT$;" "I;Z4$
I; FOR "I;ZK$;"("I;Z7$;"I;IES:IF P THEN PR
INT#3," ":PRINT#3,PT$;" "I;Z4$;" FOR "I;ZK
$;"("I;Z7$;"I;IES
1040 DT#=#0:CT#=#0:BL#=#0:GOSUB520
1050 FORI=1TOW
1060 IFMID$(A$(I),26,2)<KAK$THEN1110
1070 IF E$="S" THEN1090
1080 IF E$<>LEFT$(A$(I),6) THEN1110
1090 GOSUB1570:GOSUB1620:PRINTVX$;IF P T
HEN PRINT#3,VX$
1100 DR#=#0:VAL(MID$(A$(I),31,8)):CR#=#0:VAL(M
ID$(A$(I),39,8)):DT#=#0:DR#=#0:CT#=#0:CR#
:BL#=#0:DR#=#0:CR#
1110 NEXT
1120 PRINT:PRINTZ0$;TAB(36):PRINTUSINGF
3$;DT#;CT#;PRINT"BALANCE "I;PRINTUSINGF4
$;BL#
1130 IF P THEN PRINT#3," ":PRINT#3,Z0$;
TAB(36):PRINT#3,USINGF3$;DT#;CT#;PRINT#
3," BALANCE "I;PRINT#3,USINGF4$;BL#;CLOSE
#3
1140 LOCATE0,15:PRINT"PRINTOUT COMPLETE
- "I;S0$;PA=1006:LN=1:GOSUB60:GOTO740
1150 GOSUB230:AD$="Y":GOSUB1000:LOCATE8,
3:PRINT"LINEPRINTER UTILITY":LOCATE0,5:P
RINT"TYPE HEADINGS OR NOTES AS REQUIRED"
:LOCATE0,6:PRINT"TYPE "I;CHR$(34); "EXIT" I
CHR$(34); " TO RETURN TO MAIN MENU"
1160 M$="":INPUTM$:IFM$="EXIT" THEN CLOSE
#3:GOTO250
1170 PRINTM$:PRINT#3,M$:GOTO1160

```



```

330 L= LEN(LE$): IF RI$="" THEN R= -1 EL
SE R= LEN(RI$)
340 FOR I=1 TO VB: IF L > LEN(A$(I)) THEN
360
350 IF LE$ < LEFT$(A$(I),L) THEN 360 EL
SE 380
360 NEXT I
370 IF C$ < > THEN PRINT "I don't unders
tand "CHR$(34);C$;CHR$(34)", check my vo
cabulary.": GOTO 270
380 IF R= -1 THEN 420
390 FOR J=1 TO ND
400 IF RI$ < >B$(J) THEN NEXT J ELSE 420
410 PRINT "I don't understand "CHR$(34);R
I$;CHR$(34)", check my vocabulary.": GOTO
0 270
420 ON I GOSUB 450,450,450,450,830,880,8
80,880,950,950,950,950,980,1050,980,1230
,980,1120,1160,1290,1360,2010
430 IF I>4 AND I<13 THEN 210
440 IF I=22 THEN 180 ELSE 170
450 IF J<OB+1 THEN PRINT "I can't "CHR$(13
4);A$(I)+ " "RI$;CHR$(34)".": GOTO 270
460 J=J-OB: ON J GOTO 550,640,710,750,55
0,640,710,750,470,490,510,530,470,490,51
0,530,790,810,790,810
470 IF L=13 THEN L=11 ELSE GOSUB 1860
480 RETURN
490 IF L=12 THEN L=11 ELSE IF L=14 TH
EN L=15 ELSE GOSUB 1860
500 RETURN
510 IF L=11 THEN L=12 ELSE IF L=15 TH
EN L=14 ELSE GOSUB 1860
520 RETURN
530 IF L=11 THEN L=13 ELSE GOSUB 1860
540 RETURN
550 IF L=2 THEN L=1 ELSE IF L=5 THEN
L=4 ELSE IF L=6 THEN L=5
560 IF L=7 THEN L=9 ELSE IF L=11 THEN
L=7
570 IF L=16 AND B(4)= -1 THEN GOSUB 187
0
580 IF L=16 AND B(4) < > -1 THEN L=17
590 IF L=18 AND B(5)= -1 THEN L=19
600 IF L=18 AND B(5) < > -1 THEN GOSUB 1
870
610 IF L=15 THEN L=16
620 IF L=OL THEN GOSUB 1860
630 RETURN
640 IF L=1 THEN L=2 ELSE IF L=4 THEN
L=5 ELSE IF L=5 THEN L=6
650 IF L=9 THEN L=7 ELSE IF L=7 THEN
L=11 ELSE IF L=16 THEN L=15
660 IF L=17 THEN L=16
670 IF L=19 AND B(5)= -1 THEN L=18
680 IF L=19 AND B(5) < > -1 THEN GOSUB 1
870
690 IF L=OL THEN GOSUB 1860
700 RETURN
710 IF L=3 THEN L=2 ELSE IF L=4 THEN
L=3 ELSE IF L=10 THEN L=7

```

\*\*\*\* SIRIUS ADVENTURE \*\*\*\*

HITACHI PEACH

```

10 REM SIRIUS ADVENTURE
20 REM (C) MAY 1983 MLADEN BAUK.
30 REM
40 REM MODIFIED FOR THE HITACHI PEACH
BY -- MICRO-80
60 CLS:SCREEN0,,0
70 LOCATE7,4:PRINT " S I R I U S ":LOCAT
E7,5:PRINT " ADVENTURE":FOR X=1TO2000:N
EXT:CLS
80 CLEAR 200: VB=22: ND=26: L=21: OB=6:
LN=337
90 CLS: LOCATE15,3:PRINT "Sirius Adventur
e": PM$="": PF$=" "
100 LOCATE9,5:PRINT "Press: <I> nstructi
ons or"
110 LOCATE17,6:PRINT "<B> egin.": CL$=""
120 DIM A$(VB), B$(ND), L$(L), B(OB): GO
SUB 1430
130 A$= INKEY$: IF A$="" THEN 130
140 IF A$="I" THEN 1720
150 IF A$ < >B" THEN 130
160 CLS
170 IF L=OL THEN 270
180 OL=L:CLS: LOCATE15,3:PRINT "Sirius A
dventure"
190 IF L>4 AND B(1) < > -1 THEN PRINT: P
RINT " It's too dark to see!": GOTO 2
70
200 PRINT "I am ...":PRINT L$(LO)
210 LINE (0,100)-(656,100),PSET
220 TR=0: LOCATE7,20:PRINT CL$: LOCATE7,
20:PRINT "Visible objects >>> "
230 FOR I=1 TO OB
240 IF B(I)=LO THEN PRINT B$(I);". ": I
R= -1
250 NEXT I
260 IF TR < > -1 THEN LOCATE27,20:PRINT "N
one.":
270 LOCATE0,15:PRINT "What should I do?";
CL$: C$="":GOSUB 1900: PRINT: PRINT
280 IF C$="" THEN PRINT "Huh?": GOTO 270
290 FOR I=1 TO LEN(C$): IF ASC( MID$( C$
, I, 1))=32 THEN 310
300 NEXT I: GOTO 320
310 IF$= LEFT$( C$, I-1): RI$= MID$( C$,
I+1, LEN(C$)- LEN(IF$)-1):GOTO 330
320 LE$= LEFT$(C$, I): RI$=""

```

```

1490 FOR I=1 TO 08: READ B(I): NEXT I
1500 FOR I=1 TO L: READ L$(I): NEXT I: R
RETURN
1510 DATA "at a plateau near a cliff. A
rocky path leads south. Some obvious ex
its: South."
1520 DATA "on a rocky path leading north
and curving to the east. Some obv
ious exits: North. East."
1530 DATA "at the entrance to a dark cav
e. A rocky path to the west curves north
. There is a slight breeze. Some obvious
exits: West. East."
1540 DATA "just inside a dark cave. Ligh
t comes from an entrance to the west.
There is a dank, mouldy smell. A tunnel l
eads south. Some obvious exits: West. Sou
th."
1550 DATA "in a low north/south tunnel.
Some obvious exits: North. South."
1560 DATA "in an oval cavern. There is a
forbidding stone staircase here. Some ob
vious exits: North. Down."
1570 DATA "in a high, square cave with w
alls of frozen ice. There are passage
s in many directions. Some obvious exit
s: North. South. West. East. Up."
1580 DATA "in a triangular side-chamber.
Some obvious exits: East."
1590 DATA "in a musty-smelling alcove. S
ome obvious exits: South."
1600 DATA "in an eerie chamber - small s
quealing sounds come from the walls. S
ome obvious exits: West."
1610 DATA "in an enormous cave. There is
a double pillar of green stone down th
e centre. Some obvious exits: North. So
uthwest. Southeast."
1620 DATA "in a malodorous tunnel. Some
obvious exits: Northeast."
1630 DATA "in a room in which the only v
isible exit is the way I came in. Some ob
vious exits: Northwest"
1640 DATA "in a secret room, reached only
by magical means. Some obvious exits: No
rtheast."
1650 DATA "in an octagonal room. Some obv
ious exits: North. Southwest."
1660 DATA "in an enormous misty cavern.
Mist obscures the ceiling. Some ob
vious exits: North. South."
1670 DATA "in a tiny box-shaped room. Do
or leads south and stairs lead down. S
ome obvious exits: South. Down."
1680 DATA "in a strange room. There's a
faint whiff of chlorine. Some obvious exi
ts: North. Up."
1690 DATA "in a steamy chamber, with war
m walls. Some obvious exits: West. Sou
th."

```

```

1080 PRINT "The room dims and blurs, and.
.";
1090 FOR I=1 TO 1000: NEXT I
1100 IF LO=13 THEN LO=14 ELSE IF LO=14 T
HEN LO=13 ELSE PRINT "nothing happens.":
RETURN
1110 PRINT "I am magically transported!":
FOR I=1 TO 1000: NEXT I: RETURN
1120 PRINT "Confirm <Y/N> ?": C$="": LO
CATE$,3:PRINTN,CL$: GOSUB 1900
1130 IF C$="Y" THEN A=USR1(0):CLS: END
1140 IF C$<"N" THEN 1120
1150 PRINT:PRINT "Confirm <CANCEL
L>": RETURN
1160 IN=0: FOR I=4 TO 6
1170 IF B(I)=1 THEN IN=IN+20
1180 NEXT I
1190 IF IN=60 THEN PRINT "Fantastic!
You have solved the adventure!"
1200 PRINT "You have IN points out of a p
ossible 60."
1210 IF IN=60 THEN A=USR1(0):CLS: END
1220 RETURN
1230 PRINT "I am carrying >>> ";
1240 IN=0: FOR I=1 TO 08
1250 IF B(I)= -1 THEN PRINT "A B$(I):"
.": IN= -1
1260 NEXT I
1270 IF IN< -1 THEN PRINT "Nothing at a
11.": RETURN
1280 RETURN
1290 PRINT "Ready tape...press <ENTER>"
1300 IN= INKEY$: IF IN$="" THEN 1370
1310 OPEN "O",#1,"CAS0:URLORD"
1320 FOR I=1 TO 08: PRINT#1,B(I): NEX
T I
1330 PRINT#1,LO
1340 CLOSE #1
1350 RETURN
1360 PRINT "Ready tape...press <ENTER>"
1370 IN= INKEY$: IF IN$="" THEN 1370
1380 OPEN "I",#1,"CAS0:URLORD"
1390 FOR I=1 TO 08: INPUT#1,B(I): NEXT
I
1400 INPUT#1,LO
1410 CLOSE #1
1420 RETURN
1430 LO=1
1440 FOR I=1 TO VB: READ A$(I): NEXT I
1450 FOR I=1 TO ND: READ B$(I): NEXT I
1460 DATA GO,WALK,RUN,CRAWL,EAT,GET,TAKE
,GRAB,DROP,THROW,PUT,LEAVE,LOOK,WAVE,EXA
MINE,INVENTORY,INSPECT,QUIT,SCORE,SAVE,L
OAD,VOCABULARY
1470 DATA LAMP,BUN,ROD,RING,STATUE,CROWN
,N,S,W,E,NORTH,SOUTH,WEST,EAST,NW,NE,SW,
SE,NORTHWEST,NORTHEAST,SOUTHWEST,SOUTHEA
ST,UP,DOWN,U,D
1480 DATA 1,6,9,8,12,21

```

```

1700 DATA "in a large room, littered wit
h alabaster slabs. Some obvious exits: We
st. East."
1710 DATA "in the throne room of the evil
Ur lord! A low door leads east. Some o
bvious exits: East."
1720 CLS:LOCATE7,3:PRINT":PRINT>Your q
uest is to explore the cave of the";
1730 PRINT"evil Ur lord, and bring back t
o the edge of the cliff the following va
luables:"
1740 PRINT"1. The white gold ring."
1750 PRINT"2. The sacred silver statue."
1760 PRINT"3. The jewelled crown of the
Ur lord.";
1770 PRINT:PRINT
1780 PRINT"Be careful...":PRINT:PRINT:PR
INT
1790 PRINT"Press <C> continue."
1800 FOR I=1 TO 4000
1810 A$= INKEY$: IF A$="" THEN 1840
1820 IF A$<"C" THEN 1810
1830 GOTO 170
1840 NEXT I
1850 GOTO 170
1860 PRINT"You cannot go in that directi
on.": RETURN
1870 PRINT"An invisible force prevents y
ou from passing."
1880 FOR I=1 TO 1000: NEXT I
1890 RETURN
1900 LOCATE18,15:PRINTC$;PM$;" ";
1910 A$= INKEY$: IF A$="" THEN 1910
1920 LOCATE18,15:PRINT,PF$; A=ASC(A$)
1930 IF A>31 THEN 1990
1940 IF A=8 AND LEN(C$)>0 THEN C$= LEFT$
(C$,LEN(C$)-1): LOCATE18,15:PRINTCL$:LO
CATE18,15:PRINTC$; GOTO 1900 ELSE IF A
=8 THEN 1900
1950 IF A=13 THEN GOSUB2040:FOR X=15 TO
23:LOCATE0,X:PRINT"
":NEXT:LOCATE0,15:RETU
RN
1960 IF A=10 THEN A$= CHR$(92) ELSE IF A
=27 THEN A$="@"
1970 IF A=9 THEN A$= CHR$(187) ELSE IF A
=31 THEN A$="X"
1980 IF A=24 THEN C$="": LOCATE18,15:PRI
NTCL$: GOTO 1900
1990 C$=C$+A$: IF LEN(C$)>20 THEN RETURN
2000 LOCATE18,15:PRINTC$; GOTO 1900
2010 CLS:LOCATE15,3:PRINTA$(22):LOCATE7,
4:PRINT;
2020 FOR I=1 TO VB: PRINT A$(I):, NEXT
I
2030 A$= INKEY$: IF A$="" THEN 2030 ELSE
RETURN
2040 X=(PEEK(39)*256+PEEK(40))-(PEEK(33)
*256+PEEK(34)):RETURN

```

```

520 PRINT@244,CHR$(249);:PRINT@2
77,CHR$(175);:PRINT@308,CHR$(246
);
530 PRINT@457,"SCORE ";M;:PRINT@
11,"ROUND ";R;:X=275:E=1:S=S-10:
A$="":FOR T=1 TO 40:A$=A$+CHR$(
RND(7)*16+143):NEXT:A$=A$+STRING
$(20,128)
540 IF R<5 THEN PRINT@480,B$;:G=
2:SV=S/2-5 ELSE G=1:SV=S
550 REM PRINT SHIPS LEFT
560 PRINT@496+(5-SH)*3,LEFT$(SH$
,3*SH);:GOTO 230
570 REM SHIP DESTROYED
580 FOR Z=1 TO 10:PRINT@258,CHR$(
RND(7)*16+RND(14)+128);:SOUND12
0-Z*10,1:FOR ZZ=1 TO 50:NEXT:NEX
T:SH=SH-1:S=S+10:R=R-1:S1=S1-1
590 IF SH<1 THEN 610 ELSE RETURN
600 REM YOU LOST
610 CLS0:PRINT@96,"all"CHR$(128)
"your"CHR$(128)"ships"CHR$(128)"
were"CHR$(128)"destroyed";:PRINT
@128,"before"CHR$(128)"the"CHR$(
128);
615 PRINT"satellite"CHR$(128)"ce
ased"CHR$(128)"to";:PRINT@160,"f
unction";:PRINT@230,"earth"CHR$(
128)"was"CHR$(128)"destroyed";
620 A=359
630 REM FINAL SCORE
640 PRINT@A,"your"CHR$(128)"scor
e"CHR$(128);M;
650 REM NEXT GAME
660 PRINT@500,"continued";
670 0$=INKEY$
680 IF INKEY$="" THEN 680
690 CLS0:PRINT@106,"ANOTHER GAME
?";
700 0$=INKEY$:IF 0$="" THEN 700
ELSE IF 0$="Y" THEN 200 ELSE I
F 0$="N" THEN END ELSE 700
710 REM YOU'VE WON
720 CLS0:PRINT@170,"congratulati
ons";:PRINT@234,"earth"CHR$(128)
"was"CHR$(128)"saved";:A=361:GOF
0640

```

```

170 IF INKEY$="" THEN IF T THENP
LAY"01L10CP8":SCREEN0,1 ELSE PLA
Y"01L10CP8":SCREEN0,0 ELSE 190
180 T=NOT(T):GOTO 170
190 REM INITIALIZE VARIABLES
200 CLS0:X=270:S=110:SH=3:E=1:S1
=0:M=0:R=0
210 GOTO500
220 REM CONTROL LOOP
230 GOSUB 430
240 REM FIND COLOUR MATCH
250 F=0
260 F=INSTR(F+1,0$,D$):IF F=0 TH
EN 290
270 MID$(A$,F)=MID$(A$,F+1)
280 X=X+1:M=M+10:E=E-1
290 0$=MID$(A$,1,E)
300 REM UPDATE DISPLAY
310 PRINT@X,0$;:PRINT@463,M;
320 PLAY"L2001C"
330 REM CHECK FOR WIN
340 IF LEFT$(A$,1)=CHR$(128) THE
N M=M+100#R:GOTO 500
350 GOSUB 430
360 REM CHECK FOR LOSS
370 X=X-1:E=E+1:IF X<259 THEN M
=M-400:GOSUB580:GOTO500
380 GOSUB 430
390 GOTO230
400 REM END CONTROL LOOP
410 REM
420 REM CHANGE SHIP COLOUR
430 FOR V=1 TO 6
440 IF PEEK(345)=247 THEN C=C+1:
IF C>7 THEN C=1
450 D$=CHR$(143+16*C):PRINT@258,
D$;
460 FOR I=1 TO SV:NEXT
470 NEXT V
480 RETURN
490 REM NEXT LEVEL
500 R=R+1:S1=S1+1:IF R>11 THEN 7
20 ELSE CLS0:FOR Z=1 TO 10:X1=RN
D(64)-1:Y1=RND(32)-1:SET(X1,Y1,5
):NEXT
510 IF S1>5 THEN S1=2:SH=SH+1:I
F SH>5 THEN SH=5

```

\*\*\*\* KILLER SATELLITE \*\*\*\*

COLOUR COMPUTER

```

10 REM THE KILLER SATELLITE
20 REM VERSION 1.1
30 REM 15/8/83
40 REM ** BY N.S. EDWARDS **
50 REM INITIAL DISPLAY
60 CLS0:FOR T=1 TO 15:SET(RND(64
)-1,RND(32)-1,5):NEXT:PRINT@262,
"the"CHR$(128)"killer"CHR$(128)"
satellite";
70 PRINT@340,CHR$(249);:PRINT@37
3,CHR$(175);:PRINT@404,CHR$(246
);
80 REM SET UP STRINGS
90 FOR T=1 TO 5:SH$=SH$+CHR$(183
)+CHR$(187)+CHR$(128):NEXT
100 FOR T=1 TO 7:B$=B$+CHR$(143+
16*T):NEXT
110 PRINT@365,B$;
120 FOR T=1 TO 8:PLAY"01L20CP10"
:NEXT
130 CLS:PRINT"A SATELLITE ORBITI
NG FAR ABOVE THE EARTH HAS MALF
UNCTIONED. IT HAS BEGUN SENDI
NG DANGEROUS COSMIC";
135 PRINT" RAYS TOWARDS THE EART
H. YOU MUST PROTECT THE EARTH
FROM THE COSMIC RAYS BY MATC
HINGTHE COLOURS OF THE APPROACHI
NG RAYS";
140 PRINT" WITH YOUR SHIELD UNTI
L THE SATELLITE RUNS OUT OF POWE
R (11 ROUNDS).":PRINT:PRINT"TO D
O THIS USE THE <SPACE BAR> .":PR
INT" BE WARNED THE RAYS TRAVEL
AT EVER INCREASING SPEEDS."
150 PRINT@484,"PRESS <ENTER> TO
BEGIN";
160 T=-1

```



```

**** SIRIUS ADVENTURE ****
COLOUR COMPUTER

10 REM      SIRIUS ADVENTURE
20 REM (C) MAY 1983 MLADEN BAUK.

30 REM      HI-RES WRITER 1.1
      (C) 1983 G.D. WILLIAMSON

40 REM MODIFIED FOR THE COLOUR
    COMPUTER BY -- MICRO-80

50 A=PEEK(&H1B)*256+PEEK(&H1C)-&
H4CD
60 DEFUSR0=A:DEFUSR1=A+&H1D:A=US
R0(0):PMODE2,1:PCLS:SCREEN1,1
70 PRINT064," S I R I U S ":"PRI
NT0128," ADVENTURE":FOR X=1TO2
000:NEXT:PMODE4,1:PCLS:SCREEN1,1
80 CLEAR 200:VB=22:ND=26:L=21
:OB=6:LN=337
90 CLS:PRINT08,"Sirius Adventur
e":PW0="":PF0="
100 PRINT0130,"Press: <I> nstru
ctions or"
110 PRINT0170,"<B> egin.":CL0="
120 DIM A$(VB),B$(ND),L$(L),B
(OB):GOSUB 1430
130 A$=INKEY$:IF A$="" THEN I3
0
140 IF A$="I" THEN I720
150 IF A$<"B" THEN I30
160 PCLS
170 IF L0=0L THEN 270
180 OL=L0:PCLS:PRINT08,"Sirius
Adventure"
190 IF L0>4 AND B(1)<>-1 THEN
PRINT:PRINT" It's too dark
to see!":GOTO 270
200 PRINT"I am ...":PRINTL$(L0)
210 LINE(0,100)-(256,100),PSET
220 TR=0:PRINT0448,CL0:PRINT0
448,"Visible objects >>>"
230 FOR I=1 TO OB
240 IF B(1)=L0 THEN PRINTB$(I):
:TR=-1
250 NEXT I
260 IF TR<>-1 THEN PRINT0468,"
None.":
270 PRINT0320,"What should I do?
":CL0="":GOSUB 1900:PRINT:
PRINT
280 IF C$="" THEN PRINT"Huh?":G
OTO 270
290 FOR I=1 TO LEN(C$):IF ASC(
MID$(C$,I,1))=32 THEN 310
300 NEXT I:GOTO 320

```

```

310 LE$=LEFT$(C$,I-1):RI$=M
ID$(C$,I+1,LEN(C$)-LEN(LE$)-
1):GOTO 330
320 LE$=LEFT$(C$,I):RI$=""
330 L=LEN(LE$):IF RI$="" THEN
R=-1 ELSE R=LEN(RI$)
340 FOR I=1 TO VB:IF L>LEN(A$(
I)) THEN 360
350 IF LE$>LEFT$(A$(I),L) THE
N 360 ELSE 380
360 NEXT I
370 IF C$>" THEN PRINT"I don'
t understand "CHR$(34);C$;CHR$(3
4)", check my vocabulary.":GOTO
270
380 IF R=-1 THEN 420
390 FOR J=1 TO ND
400 IF RI$>B$(J) THEN NEXT J E
LSE 420
410 PRINT"I don't understand "CH
R$(34);RI$;CHR$(34)", check my v
ocabulary.":GOTO 270
420 ON I GOSUB 450,450,450,450,8
30,880,880,880,950,950,950,950,9
80,1050,980,1230,980,1120,1160,1
290,1360,2010
430 IF I>4 AND I<13 THEN 210
440 IF I=22 THEN 180 ELSE 170
450 IF J<OB+1 THEN PRINT"I can't
"CHR$(34);A$(I)+":RI$;CHR$(34
)":GOTO 270
460 J=J-OB:ON J GOTO 550,640,71
0,750,550,640,710,750,470,490,51
0,530,470,490,510,530,790,810,79
0,810
470 IF L0=13 THEN L0=11 ELSE GOS
UB 1860
480 RETURN
490 IF L0=12 THEN L0=11 ELSE IF
L0=14 THEN L0=15 ELSE GOSUB 1860
500 RETURN
510 IF L0=11 THEN L0=12 ELSE IF
L0=15 THEN L0=14 ELSE GOSUB 1860
520 RETURN
530 IF L0=11 THEN L0=13 ELSE GOS
UB 1860
540 RETURN
550 IF L0=2 THEN L0=1 ELSE IF L0
=5 THEN L0=4 ELSE IF L0=6 THEN L
0=5
560 IF L0=7 THEN L0=9 ELSE IF L0
=11 THEN L0=7
570 IF L0=16 AND B(4)=-1 THEN G
OSUB 1870
580 IF L0=16 AND B(4)<>-1 THEN
L0=17
590 IF L0=18 AND B(5)=-1 THEN L
0=19
600 IF L0=18 AND B(5)<>-1 THEN
GOSUB 1870

```

```

610 IF L0=15 THEN L0=16
620 IF L0=0L THEN GOSUB 1860
630 RETURN
640 IF L0=1 THEN L0=2 ELSE IF L0
=4 THEN L0=5 ELSE IF L0=5 THEN L
0=6
650 IF L0=9 THEN L0=7 ELSE IF L0
=7 THEN L0=11 ELSE IF L0=16 THEN
L0=15
660 IF L0=17 THEN L0=16
670 IF L0=19 AND B(5)=-1 THEN L
0=18
680 IF L0=19 AND B(5)<>-1 THEN
GOSUB 1870
690 IF L0=0L THEN GOSUB 1860
700 RETURN
710 IF L0=3 THEN L0=2 ELSE IF L0
=4 THEN L0=3 ELSE IF L0=10 THEN
L0=7
720 IF L0=7 THEN L0=8 ELSE IF L0
=19 THEN L0=20 ELSE IF L0=20 THE
N L0=21
730 IF L0=0L THEN GOSUB 1860
740 RETURN
750 IF L0=2 THEN L0=3 ELSE IF L0
=3 THEN L0=4 ELSE IF L0=7 THEN L
0=10
760 IF L0=8 THEN L0=7 ELSE IF L0
=20 THEN L0=19 ELSE IF L0=21 THE
N L0=20
770 IF L0=0L THEN GOSUB 1860
780 RETURN
790 IF L0=7 THEN L0=6 ELSE IF L0
=18 THEN L0=17 ELSE GOSUB 1860
800 RETURN
810 IF L0=6 THEN L0=7 ELSE IF L0
=17 THEN L0=18 ELSE GOSUB 1860
820 RETURN
830 IF J=0 THEN J=3
840 IF J<>2 THEN PRINT"I can't
eat that, stupid.":RETURN
850 IF J=2 AND B(J)=0 THEN PRINT
"I already ate it.":RETURN
860 IF J=2 THEN PRINT"Munch, cho
mp, <BURP> - the creambun was de
licious!":B(2)=0:RETURN
870 PRINT"ERROR":STOP
880 IF J>OB THEN PRINT"I can't "
CHR$(34);C$;CHR$(34)".":RETURN
890 IF B(J)=-1 THEN PRINT"I air
eady have it!":RETURN
900 IF B(J)<>L0 THEN PRINT"I ca
n't see the "PRINTB$(J)" here."
:RETURN
910 IT=1:FOR I9=1 TO OB:IF 0:
9)= -1 THEN IT=IT+1:NEXT I9 ELS
E NEXT I9
920 IF IT>3 THEN PRINT"I am carr
ying too much, check inventory
":RETURN

```

```

930 PRINT"OK. I add a "B$(J):PR
INT"to my inventory."
940 B(J)= -1: RETURN
950 IF J>0B THEN PRINT"I can't "
CHR$(34);CHR$(34)".": RETURN
960 IF B(J)< > -1 THEN PRINT"I d
on't have a "R$: RETURN
970 B(J)=LO: PRINT"OK": RETURN
980 IF J>0B THEN PRINT"I don't s
ee anything special.": RETURN
990 IF B(J)< > -1 THEN PRINT"I a
m not carrying a "B$(J): RETURN
1000 ON J GOTO 1010,1020,1030,10
30,1030,1040
1010 PRINT"It burns brightly.":
RETURN
1020 PRINT"It looks tasty.": RET
URN
1030 PRINT"Magic seems to emanat
e from the "B$(J): RETURN
1040 PRINT"Its beautiful!": RETU
RN
1050 IF J>0B THEN PRINT"You are
being silly.": RETURN
1060 IF B(J)< > -1 THEN PRINT"I
don't have the "B$(J)".": RETURN
1070 IF J< > 3 THEN PRINT"Waving
the "B$(J):PRINT" is not very re
warding.": RETURN
1080 PRINT"The room dims and blu
rs, and...";
1090 FOR I=1 TO 1000: NEXT I
1100 IF LO=13 THEN LO=14 ELSE IF
LO=14 THEN LO=13 ELSE PRINT"not
hing happens.": RETURN
1110 PRINT"I am magically transp
orted!": FOR I=1 TO 1000: NEXT I
: RETURN
1120 PRINT"Confirm <Y/N> ?": C
$==: PRINTLN,CL$: GOSUB 1900
1130 IF C$="Y" THEN A=USR1(0):CL
S: END
1140 IF C$< >"N" THEN 1120
1150 PRINT:PRINT"Confirm
<CANCELLED>": RETURN
1160 IN=0: FOR I=4 TO 6
1170 IF B(19)=1 THEN IN=IN+20
1180 NEXT I
1190 IF IN=60 THEN PRINT"Fantast
ic! you have solved the adventu
re!"
1200 PRINT"You have "IN"points ":
PRINT"out of a possible 60."
1210 IF IN=60 THEN A=USR1(0):CLS
: ENN
1220 RETURN
1230 PRINT"I am carrying >>> ";
1240 IN=0: FOR I=1 TO 0B
1250 IF B(19)= -1 THEN PRINT"A "
B$(I);". ": IN= -1
1260 NEXT I
1270 IF IN< > -1 THEN PRINT"North
ing at all.": RETURN
1280 RETURN
1290 PRINT"Ready tape...press <E
NTER>."
1300 IN$= INKEY$:IF IN$=" " THEN 1300
1310 OPEN"0",#-1,"URL02"
1320 FOR I=1 TO 0B: PRINT#-1,B(
I):NEXT I
1330 PRINT#-1,LO
1340 CLOSE#-1
1350 RETURN
1360 PRINT"Ready tape...press <E
NTER>."
1370 IN$= INKEY$:IF IN$=" " THEN 137
0
1380 OPEN"1",#-1,"URL02"
1390 FOR I=1 TO 0B: INPUT#-1,B(I
9):NEXT I
1400 INPUT#-1,LO
1410 CLOSE#-1
1420 RETURN
1430 LO=1
1440 FOR I=1 TO 0B: READ A$(I):
NEXT I
1450 FOR I=1 TO 0B: READ B$(I):
NEXT I
1460 DATA GO,WALK,RUN,CRAWL,EAT,
GET,TAKE,GRAB,DROP,THROW,PUT,LEA
VE,LOOK,WAVE,EXAMINE,INVENTORY,I
NSPECT,QUIT,SCORE,SAVE,LOAD,VOCA
BULARY
1470 DATA LAMP,BUN,ROD,RING,STAT
UE,CROWN,N,S,W,E,NORTH,SOUTH,WES
T,EAST,NW,NE,SW,SE,NORTHWEST,NOR
THEAST,SOUTHWEST,SOUTHEAST,UP,DO
WN,U,D
1480 DATA 1,6,9,8,12,21
1490 FOR I=1 TO 0B: READ B(I): N
EXT I
1500 FOR I=1 TO L: READ L$(I): N
EXT I: RETURN
1510 DATA "at a plateau near a c
liff. A rocky path leads so
uth. Some obvious exits: South.
".
1520 DATA "on a rocky path leadin
g north and curving to the ea
st. Some obvious exits: North.
East."
1530 DATA "at the entrance to a
dark cave. A rocky path to the w
est curves north. There is a sli
ght breeze. Some obvious exits: W
est. East."
1540 DATA "just inside a dark ca
ve. Light comes from an entranc
e to the west. There is a dan
k, mouldy smell. A tunnel lead
s south. Some obvious exits: W
est. South."
1550 DATA "in a low north/south
tunnel. Some obvious exits: N
orth. South."
1560 DATA "in an oval cavern.
There is a forbidding stone stal
e case here. Some obvious exits: N
orth. Down."
1570 DATA "in a high, square cave
with walls of frozen ice. There a
re passages in many directions. So
me obvious exits: North. South. We
st. East. Up."
1580 DATA "in a triangular side-
chamber. Some obvious exits: E
ast."
1590 DATA "in a musty-smelling a
lcove. Some obvious exits: South.
".
1600 DATA "in an eerie chamber -
small squealing sounds come
from the walls. Some obvious ex
its: West."
1610 DATA "in an enormous cave.
There is a double pillar of gr
een stone down the centre. Some
obvious exits: North. Southwe
st. Southeast."
1620 DATA "in a malodorous tunn
el. Some obvious exits: Northe
ast."
1630 DATA "in a room in which th
e only VISIBLE exit is the w
ay I came in. Some obvious exits
: Northwest"
1640 DATA "in a secret room, reac
hed only by magical means. Some ob
vious exits: Northeast."
1650 DATA "in a octagonal room. S
ome obvious exits: North. Southwe
st."
1660 DATA "in an enormous misty
cavern. Mist obscures the cei
ling. Some obvious exits: North.
South."
1670 DATA "in a tiny box-shaped
room. Door leads south and stair
s lead down. Some obvious exits: S
outh. Down."
1680 DATA "in a strange room. I
here is a faint whiff of chlori
ne. Some obvious exits: North.
Up."

```

```

2300 IFPEEK(K2)=251 THENPRINT@GP,B
L$:=GP+1:POKEK2,0:IFGP=510THE
NGP=509
2400 IFGOTHENIFPEEK(K3)=247 THENM=
GP-480:M=M*2:M=M+1:GO=0:F=-1:POK
EK3,0
2500 POKE17023,0:SOUND100,1:SOUND
260,1
2600 IFFTHENY=Y-1:IFY<KK+2 THENGOS
UB310
2700 IFFTHENSET(M,Y,1):RESET(M,Y+
1)
2800 IFY=KKT HENG0=-1:F=0:Y=29
2900 GOT0170
3000 TF#=CHR$(137+0)+CHR$(136+0):
RETURN
3100 S=POINT(M,Y-1)
3200 ONS+1GOT0330,340,350,360,370
,380,390,410
3300 RETURN
3400 V1$=BL$:GOT0430
3500 V2$=BL$:GOT0430
3600 V3$=BL$:GOT0430
3700 V4$=BL$:GOT0430
3800 V5$=BL$:GOT0430
3900 V6$=BL$:GOT0430
4000 CLS(0)
4100 V7$=BL$
4200 F=0:GO=-1
4300 GOSUB150:SOUND20,2:SOUND10,4
:SC=SC+10:IFSC=70 THENGOT0450
4400 RETURN
4500 CLS:PRINT"YOU WIN. YOUR SCOR
E WAS 100":GOT0470
4600 CLS:PRINT"YOU LOST. YOUR SCO
RE WAS ";SC
4700 PRINT"PLAY AGAIN (Y/N)"
4800 IN$=INKEY$:IFIN$="" THEN480
4900 IFIN$="Y" THENRUN300
5000 CLS:PRINT"BYE"
5100 END
5200 CLS:PRINT"***** MC 10 MICRO
VADERS *****"
5300 PRINT:PRINT"SHOOT DOWN ALL 0
F THE INVADERS BEFORE THEY LAND
TO WIN. 10 POINTS FOR EA
CH INVADER THAT"
5400 PRINT"IS DESTROYED. A BONUS
OF 30 POINTS IS AWARDED FOR
DESTROYINGALL THE INVADERS. MAXI
MUM SCORE IS 100"
5500 PRINT:PRESS<A> TO MOVE LEFT
PRESS<S> TO MOVE RIGH
T PRESS THE SPACE BAR TO
FIRE"
5600 PRINT:PRINT"PRESS ANY KEY TO
BEGIN"
5700 IN$=INKEY$:IFIN$="" THEN570
5800 RETURN

```

```

1000 C$=C$+A$:IF LEN(C$)>20 THE
N RETURN
2000 PRINT@LN,C$;:GOTO 1900
2010 PCLS:PRINT@B,A$(22):PRINT@6
4,;
2020 FOR I9=1 TO VB:PRINT A$(I9
);:NEXT I9
2030 A$=INKEY$:IF A$="" THEN 20
30 ELSE RETURN
2040 X=(PEEK(39)*256+PEEK(40))-1
PEEK(33)*256+PEEK(34)):RETURN

**** MC-10 VADERS ****

MC-10

10 ' (C) 1/11/83
SUNBURST SOFTWARE SERVICES

20 GOSUB520
30 CLEAR500
40 OK=-1:GO=-1:Y=29
50 KK=1
60 K1=16946:K2=16948:K3=16952:PO
KEK1,0:POKEK2,0:POKEK3,0
70 CLS(0)
80 O=0:GOSUB300:V1$=TF$:O=16:GOS
UB300:V2$=TF$:O=32:GOSUB300:V3$=
TF$:O=48:GOSUB300:V4$=TF$:O=64:G
OSUB300:V5$=TF$:O=80:GOSUB300
90 V6$=TF$:O=96:GOSUB300:V7$=TF$
100 BL$=CHR$(128)+CHR$(128)
110 GN$=CHR$(247)+CHR$(242)
120 GP=490
130 P=1
140 GOSUB150:GOT0170
150 DL$=BL$+BL$+BL$+V1$+BL$+BL$+
V2$+BL$+BL$+V3$+BL$+BL$+V4$+BL$+
BL$+V5$+BL$+BL$+V6$+BL$+BL$+V7$+
BL$+BL$+BL$
160 RETURN
170 PRINT@GP,GN$:
180 I=1+1:IFI=50 THENI=0:CLS(0):K
=K+32:KK=KK+2:Y=29:F=0:GO=-1:IFK
=448 THENGOT0460
190 PRINT@K,MID$(DL$,P,32);
200 IFOK THENP=P+1:IFP+20=LEN(DL$
) THENOK=0
210 IFNOTOK THENP=P-1:IFP=1 THENOK
=-1
220 IFPEEK(K1)=254 THENPRINT@GP,B
L$;GP=GP-1:POKEK1,0:IFGP=479 THE
NGP=480

```

```

1400 DATA "in a steamy chamber,
with warm walls. Some obvious e
xits: West.South."
1700 DATA "in a large room, litt
ered with alabaster slabs. Some
obvious exits: West. East."
1710 DATA "in the throne room of
the evil Urlord! A low door l
eads east. Some obvious exits: E
ast."
1720 PCLS:PRINT@0,"";PRINT"Your
quest is to explore the cave
of the evil Urlord, and"
1730 PRINT"bring back to the edg
e of the cliff the following v
aluable:"
1740 PRINT"1. The white gold rin
g."
1750 PRINT"2. The sacred silver
statue."
1760 PRINT"3. The jewelled crown
of the Urlord.";
1770 PRINT:PRINT
1780 PRINT"Be careful...":PRINT:
PRINT:PRINT
1790 PRINT"Press <C> continue."
1800 FOR I=1 TO 4000
1810 A$=INKEY$:IF A$="" THEN 1
840
1820 IF A$<"C" THEN 1810
1830 GOTO 170
1840 NEXT I
1850 GOTO 170
1860 PRINT"You cannot go in that
direction.":RETURN
1870 PRINT"An invisible force pr
events you from passing."
1880 FOR I=1 TO 1000: NEXT I
1890 RETURN
1900 PRINT@LN+LEN(C$),PM$;
1910 A$=INKEY$:IF A$="" THEN 1
910
1920 PRINT@LN+LEN(C$),PF$;A=A$
C(A$)
1930 IF A>31 THEN 1990
1940 IF A=8 AND LEN(C$)>0 THEN C
$=LEFT$(C$,LEN(C$)-1):PRINT@LN
,CL$;PRINT@LN,C$;:GOTO 1900 EL
SE IF A=8 THEN 1900
1950 IF A=13 THEN GOSUB2040:PRIN
T@320,"";FOR Q=1 TO8:PRINT"
";:PRI
NT@320,"";NEXT: RETURN
1960 IF A=10 THEN A$=CHR$(92) E
LSE IF A=27 THEN A$="@"
1970 IF A=9 THEN A$=CHR$(187) E
LSE IF A=31 THEN A$="%"
1980 IF A=24 THEN C$="":PRINT@L
N,CL$;:GOTO 1900

```



\*\*\* L2/4K BOLD TYPE FOR THE LPVII \*\*\*

TRS-80/SYSTEM-80

```
10 LPRINT"THIS IS A ";P$="TEST":GOSUB1000
20 LPRINT" OF THE ";P$="BOLD TYPE":GOSUB1000
30 LPRINT" SUBROUTINE":END
```

999 STOP:

```
*****
* THIS SUBROUTINE PRINTS P$ IN BOLD TYPE. *
* WRITTEN BY GEORGE DAU. 19-3-83 FOR THE *
* LINE PRINTER VII. *
*****
```

1000 ZP = PEEK(16539):

IF LEN(P\$)+ZP > 80 THEN

PRINT CHR\$(13);P\$ TOO LONG.":

END

ELSE

ZS\$ = STRING\$(ZP,32):

FOR Z = 1 TO 3:

POKE 16539,Z:

LPRINT CHR\$(26);ZS\$:P\$:

NEXT Z:

POKE 16539,PEEK(16539)-1:

RETURN

\*\*\* 48K/DISK SPACE UTILITY \*\*\*

TRS-80/SYSTEM-80

```
00100 ;
00110 ;
00120 ;
00130 ;
00140 ;
00150 ;
00160 ;
00170 ;
00180 ;
00190 ;
00200 ;
00210 ;
00220 ;
00230 ;
```

SPACE/ED 48K DISK

DENNIS BARETS (C)  
286 LENNOX ST  
MARYBOROUGH,4650.

THIS PROGRAM WILL ADD SPACES TO A BASIC PROGRAM  
TO MAKE IT READABLE. IT WILL NOT ADD SPACES IF  
THERE IS ALREADY A SPACE. SPACES ARE INSERTED  
AROUND BASIC KEYWORDS.  
SINCE SOME PROGRAMS HAVE M/L AFTER REM STATEMENTS  
IT WILL NOT ADD SPACES TO A LINE AFTER IT HAS  
FOUND A REM STAT., BUT ANY SPACES ADDED BEFORE  
THE REM WILL HAVE TO BE REMOVED.

```
00240 ;
00250 ;
00260 ;
00270 ;
00280 ;
00290 ;
00300 ;
00310 ;
00320 ;
00330 ;
00340 ;
00350 ;
00360 ;
00370 ;
00380 ;
00390 ;
00400 ;
00410 ;
00420 ;
00430 ;
00440 ;
00450 ;
00460 ;
00470 ;
00480 ;
00490 ;
00500 ;
00510 ;
00520 ;
00530 ;
00540 ;
00550 ;
00560 ;
00570 ;
00580 ;
00590 ;
00600 ;
00610 ;
00620 ;
00630 ;
00640 ;
00650 ;
00660 ;
00670 ;
00680 ;
00690 ;
00700 ;
00710 ;
00720 ;
00730 ;
00740 ;
00750 ;
00760 ;
00770 ;
00780 ;
00790 ;
00800 ;
00810 ;
00820 ;
00830 ;

YOUR BASIC PROGRAM WILL STILL RUN AFTER YOU
HAVE PUT THE SPACES IN.
THIS PROGRAM SHOULD NOT BE USED TO SPACE OUT
BASIC PROGRAMS WITH VERY LONG LINES, AS THEY
MAY BECOME TOO LONG WITH THE EXTRA SPACES.
## NOTE-CMD"SPACE" DOES NOT WORK FROM NEWDOS-80,
BUT MUST BE RUN WHILE IN DOS.

TO USE FROM DISK BASIC
1/LOAD BASIC PROGRAM
2/TYPE CMD"S"
3/TYPE SPACE (TO RUN PROGRAM)
4/TYPE BASIC *
TO MOD FOR LEVEL2 CHANGE THE ORIGIN IN
LINE 290 TO SAY 7500H,AND CHANGE LINE 330
TO , DOS EQU 06CCH.

ORG 0F000H
EQU 40A4H
EQU 40F9H
EQU 1AF8H
EQU 402DH

;START OF BASIC PTR
;END OF BASIC PTR
;ADJUSTS PTRS
;RETURNS HERE AT END.

LD DE,(SBASIC)
LD HL,(EBASIC)
OR A
SBC HL,DE
INC HL
PUSH HL
POP BC
RET
CALL NBYTES
LD HL,(EBASIC)
LD DE,CUT-1
LDDR INC
INC INC
RET DI
CALL LOOP1
LD XOR
LD (COUNT),A
LD (REM),A
LD A,(DE)
LD (HL),A
LD INC
INC INC
DINZ LD
LD A,(DE)
CP 22H
JR NZ,SKIP
PUSH AF
LD A,(COUNT)
XOR 01H
LD (COUNT),A
POP AF
CP 80H
JR NC,KEYWORD

;MOVE BASIC PROG
;ZERO COUNT OF "
;ZERO REM
;MISS LINE NO. & PTR
;GET BYTE OF BAS PROG
;TEST FOR "
;COUNTS (EVEN , ODD)
;TEST FOR KEYWORD
```

```

00840 CP      3AH
00850 JR      Z,COLON
00860 CP      0H
00870 JR      Z,ENDPRG
00880 LD      (HL),A
00890 INC     HL
00900 INC     DE
00910 INC     B
00920 JR      LOOP2
00930
00940 LD      A,(COUNT)
00950 CP      0H
00960 LD      A,(DE)
00970 JR      NZ,LOOP6
00980 CP      93H
00990 JR      NZ,NOTREM
01000 LD      A,0FFH
01010 LD      (REM),A
01020 CP      95H
01030 JR      Z,ELSE
01040 LD      A,(REM)
01050 CP      0H
01060 LD      A,(DE)
01070 JR      NZ,LOOP6
01080 LD      A,B
01090 CP      0H
01100 JR      Z,SKIP2
01110 HL
01120 LD      A,(HL)
01130 CP      20H
01140 JR      Z,SPACE1
01150 INC     HL
01160 LD      (HL),20H
01170 INC     HL
01180 LD      A,(DE)
01190 LD      (HL),A
01200 INC     HL
01210 INC     DE
01220 CP      0D7H
01230 JR      NC,LOOP2
01240 CP      0BCH
01250 JR      Z,LOOP2
01260 LD      A,(DE)
01270 CP      20H
01280 JR      Z,LOOP2
01290 LD      (HL),20H
01300 INC     HL
01310 JR      LOOP7
01320
01330 DEC     HL
01340 JR      SKIP2
01350
01360 DEC     HL
01370 LD      A,(HL)
01380 CP      ,
01390 JR      Z,SPC
01400 INC     HL
01410 LD      A,(DE)
01420 LD      (HL),A
01430 INC     HL

;WANT SPACE AROUND :
;POSSIBLE END OF PRG
;STORE BYTE
;IF B=0,FIRST BYTE IN LN

;DO NOT ADD SPACE, " ODD
;REM TOKEN

;NO SPC BETWEEN : & ELSE
;ZERO IF NO REM IN LINE
;COULD BE M/L AFTER REM
;NO SPACE AT START LINE

;TEST IF SPACE BEFORE
;ALREADY SPACE BEFORE KWRD
;NO SPACE,ADD ONE
;GET KEY WORD

;NO SPACE AFTER THESE KEY
;NO SPACE BEFORE BRACKET
;NO SPACE AFTER TAB(

;TEST SPC AFTER KEYWORD

;NO SPACE, ADD SPACE

;SEE IF SPACE AT END LN
;DEL SPC(LEAVE HL AS IS)

DE
A,(DE)
0
NZ,LOOP1
(HL),A
HL
DE
A,(DE)
0H
NZ,LOOP5
(HL),A
HL
(EBCASIC),HL
(40FBH),HL
(40FDH),HL
FIXPTR
EI
JP      DOS
B,3H
LOOP8
A
(COUNT),A
KEYWD
0H
DEFB
DEFB
00H
START

SPACE DUMP
START  END  ENTRY
F000  F0CC  F01C

F000: ED 5B A4 40 2A F9 40 B7 ED 52 23 E5 C1 C9 CD 00
F010: F0 2A F9 40 11 16 F0 ED B8 23 13 C9 F3 CD 0E F0
F020: 06 04 AF 32 CB F0 32 CC F0 1A 77 23 13 10 FA 1A
F030: FE 22 20 0A F5 3A CB F0 EE 01 32 CB F0 F1 FE 80
F040: 30 0E FE 3A 28 7F FE 00 28 4B 77 23 13 04 18 DF
F050: 3A CB F0 FE 00 1A 20 F2 FE 93 20 05 3E FF 32 CC
F060: F0 FE 95 28 2D 3A CC F0 FE 00 1A 20 D0 78 FE 00
F070: 28 0A 2B 7E FE 20 28 03 23 36 20 23 1A 77 23 13
F080: FE D7 30 AB FE BC 28 A7 1A FE 20 28 A2 36 20 23
F090: 18 BB 2B 18 E7 2B 7E FE 20 28 01 23 1A 77 23 13
F0A0: 1A FE 00 C2 20 F0 77 23 13 1A FE 00 20 12 77 23
F0B0: 22 F9 40 22 FB 40 22 FD 40 CD F8 1A FB C3 2D 40
F0C0: 06 03 C3 22 F0 AF 32 CB F0 18 85 00 00

10 REM.....YAHTZEE/BAS.....
.....(c) 1982 by Tony Domigan.....
.....m/1 is entered into this line.....
20 CLS:CLEAR700:DEFINT A-Z:RANDOM=DIMP1(15),P2(15):BV=13150:C=-1

```

```

630 IF (0=1) THEN GOSUB 400 ELSE GOTO 660
640 IF (Y AND NOT Y) THEN 660 ELSE IF (Y) THEN RETURN
650 IF (Y) THEN RETURN
660 GOSUB 800: RETURN
670 S0=SV-6: ONS 800 TO 690, 720, 750, 790, 940, 1010
680 GOTO 1210
690 IF (0=3) OR (0=4) OR (0=6) THEN GOSUB 800: RETURN ELSE GOTO 700
700 IF (0=10) THEN GOSUB 400 ELSE SC=0: RETURN
710 IF (Y AND NOT Y) THEN GOSUB 800: RETURN ELSE IF (Y) THEN RETURN
720 IF (0=6) THEN GOSUB 800: RETURN ELSE IF (0=10) GOSUB 400
730 IF (Y AND NOT Y) THEN GOSUB 800: RETURN ELSE SC=0: RETURN
740 IF (Y) THEN RETURN ELSE SC=0: RETURN
750 IF (0=4) THEN SC=25: RETURN
760 IF (0=10) THEN GOSUB 400 ELSE SC=0: RETURN
770 IF (Y AND NOT Y) THEN SC=25: RETURN ELSE IF (Y) THEN RETURN
780 IF (Y) THEN RETURN ELSE SC=0: RETURN
790 IF (0=10) THEN GOSUB 400 ELSE 820
800 IF (Y AND NOT Y) THEN SC=30: RETURN ELSE IF (Y) THEN RETURN
810 IF (Y) THEN RETURN
820 IF (0<1) THEN 900 ELSE SC=0
830 W=W+1
840 IF (S(W)=S(W+1) AND W=4) THEN 900 ELSE IF (S(W)<S(W+1)) THEN 830
850 W1=S(W+1): IF (W=1) THEN S(2)=S(3): S(3)=S(4): S(4)=S(5): S(5)=W1
860 IF (W=2) THEN S(3)=S(4): S(4)=S(5): S(5)=W1
870 IF (W=3) THEN S(4)=S(5): S(5)=W1
880 FOR W=1 TO 5: PRINT 260+W*2, S(W): NEXT W
890 V1=0: V2=0
900 IF (S(1)+1=S(2) AND S(2)+1=S(3) AND S(3)+1=S(4)) THEN V1=1
910 IF (S(2)+1=S(3) AND S(3)+1=S(4) AND S(4)+1=S(5)) THEN V2=1
920 IF (V1 OR V2) THEN SC=30 ELSE SC=0
930 RETURN
940 IF (0>0) THEN GOTO 970
950 IF (1)=S(2)+CANDS(2)=S(3)+CANDS(4)=S(5)+C THEN SC=40 ELSE SC=0
960 RETURN
970 IF (0=10) THEN GOSUB 400 ELSE SC=0: RETURN
980 IF (Y AND NOT Y) THEN SC=40: RETURN ELSE IF (Y) THEN RETURN
990 IF (Y) THEN RETURN
1000 SC=0: RETURN
1010 IF (0=10) THEN SC=50: RETURN
1020 SC=0: RETURN
1030 DE=0: PRINT 257, STRING$(60, " ");
1040 PRINT 275, "**** "; Z$; " To Go ****";
1050 IF (JZ=1) THEN PRINT 338, N$;
1060 DE=DE+1: IF (DE<1) THEN 1070 ELSE SC=USR(BV+10)
1070 J$=INKEY$
1080 IF J$<>CHR$(13) THEN 1120
1090 IF (LEN(J$)=1 AND JZ=1) THEN PRINT 337, S$; : E=USR(BV): RETURN
1100 IF (LEN(H$)>0) THEN E=USR(BV): RETURN
1110 GOTO 1070
1120 IF (J$<>"") THEN E=USR(BV)
1130 IF (LEN(H$)=1) THEN PRINT 337, S$; ELSE IF (J$="") THEN 1070
1140 IF (J$<>CHR$(8)) THEN 1160
1150 IF (LEN(H$)>0) THEN H$=LEFT$(H$, LEN(H$)+C): PRINT 347, H$; " "; : G
OTO 1070 ELSE J$="": H$="": GOTO 1070
1160 H$=H$+J$: IF (LEN(H$)>7) THEN H$="": GOTO 1030 ELSE PRINT 347, H$; : G
OTO 1070
1170 IF (JZ=1) THEN PRINT 337, S$;
1180 R=ROUND(6): S(ZZ)=R: FLAG=1: GOSUB 340
1190 ON R GOTO 380, 390, 400, 410, 420, 430
1200 RETURN

```

```

30 BX=1760: BW=25550: TEST=PEEK(16548)+PEEK(16549)*256+5: GOTO 1780
40 YP=1: IF (Z$=P1$ AND P1(12)>C) OR (Z$=P2$ AND P2(12)>C) THEN 50 ELSE IF Y=1
50 IF (Z$=P1$ AND P1(11)=C AND P1(12)>C) THEN YG=1: RETURN
60 IF (Z$=P2$ AND P2(11)=C AND P2(12)>C) THEN YG=1 ELSE YG=0
70 RETURN
80 SC=S(1)+S(2)+S(3)+S(4)+S(5): RETURN
90 PRINT 752, K$; : PRINT 808, K$; : PRINT 948, K$;
100 N=N+J: IF (N=1) THEN Z$=P1$ ELSE IF (N>3 AND N<6) THEN Z$=P2$
110 PRINT 948, I$; Z$; T2$; : IF N=7 THEN N=0: GOTO 100
120 PRINT 337, S$; JZ=JZ+1: PRINT 370, "Roll No"; JZ;
130 IF (JZ=1) THEN 250 ELSE IF (JZ=3) THEN JZ=0
140 Y$="": H$="": GOSUB 1030: L$=LEFT$(H$, 1)
150 LH=LEN(H$)+C: Y$=RIGHT$(H$, LH): H$=Y$
160 IF L$="R" THEN 170 ELSE IF L$="S" THEN 1210 ELSE PRINT 337, S$; : GOTO 140
170 P=121
180 FOR J=1 TO 5: K(J)=0: NEXT J: FORDI=1 TO LEN(Y$)
190 Y1$=LEFT$(Y$, 1): Y$=RIGHT$(Y$, LEN(Y$)+C)
200 Y2$=VAL(Y1$): K(Y2)=Y2: NEXT DI
210 FOR ZZ=1 TO 5
220 IF (ZZ)<>0 THEN P=(121+(K(ZZ)*12)): GOSUB 1170
230 NEXT ZZ
240 FLAG=0: GOTO 480
250 PRINT 467, A1$; : PRINT 491, A1$;
260 GG=1: HH=6: WP=0: IF (Z$=P1$) THEN S(1)=0 ELSE S(2)=0
270 GOSUB 1490
280 GG=7: HH=13: WP=1: IF (Z$=P1$) THEN S(1)=0 ELSE S(2)=0
290 GOSUB 1490
300 P=121: FOR W=1 TO 5: P=P+12: PRINT 7, D7$; : NEXT W: GOSUB 1030
310 P=121: FOR K=1 TO 5: P=P+12
320 GOSUB 340
330 P=ROUND(6): S(K)=R: ON R GOTO 380, 390, 400, 410, 420, 430
340 GOSUB 1480: PRINT 7, D7$; : GOSUB 1480: PRINT 7, D5$;
350 GOSUB 1480: PRINT 7, D7$; : GOSUB 1480: PRINT 7, D2$;
360 GOSUB 1480: PRINT 7, D7$; : GOSUB 1480: PRINT 7, D4$;
370 GOSUB 1480: PRINT 7, D7$; : GOSUB 1480: RETURN
380 PRINT 7, D1$; : E=USR(BX): GOTO 440
390 PRINT 7, D2$; : E=USR(BX): GOTO 440
400 PRINT 7, D3$; : E=USR(BX): GOTO 440
410 PRINT 7, D4$; : E=USR(BX): GOTO 440
420 PRINT 7, D5$; : E=USR(BX): GOTO 440
430 PRINT 7, D6$; : E=USR(BX)
440 IF (K=5) THEN 450 ELSE SC=50: GOSUB 1410
450 IF (FLAG=1) THEN FLAG=0: RETURN
460 IF (ST=0) THEN RETURN
470 NEXT K
480 Q=Q+1: GOSUB 1640
490 IF (Q=3) THEN N0=0: GOTO 1210 ELSE GOTO 100
500 M=0: Q=0
510 FOR W=1 TO 4: IF (S(W)<S(W+1)) THEN 530
520 W1=S(W): S(W)=S(W+1): S(W+1)=W1: M=M+1
530 NEXT W
540 IF M THEN 500 ELSE SC=0
550 FOR F=1 TO 4: FOR G=1 TO 5: IF (S(F)=S(G)) THEN 0+1: H=S(F)
560 IF (0=6000=3000=1) THEN 1=H ELSE IF (0=2) THEN 1=H
570 NEXT G, F
580 SC=0: SV=VAL(H$): IF (SV<1) THEN 1210
590 IF (SV>0 AND SV<7) THEN 610 ELSE IF (SV=13) THEN 630
600 IF (SV<13) THEN 670 ELSE GOTO 1210
610 FOR W=1 TO 5: IF (S(W)=SV) THEN SC=SC+SV: NEXT W ELSE SC=SC: NEXT W
620 RETURN

```



```

1210 YF=0:YP=0:YG=0:PRINT@337,S$;
1220 PRINT@275,Z$;I$;E=USR(BV+35)
1230 H$="":J$="":GOSUB1070
1240 Y$=LEFT$(H$,1):IF(Y$#="R"ORY3$="S"ORY3$=CHR$(13))THEN1210
1250 GOSUB500
1260 IF(YFANDNOTY$)THENE=USR(BW):PRINT@337,F$;GOTO1770
1270 IF(YV)THENE=USR(BW):PRINT@337,"*";I$G$;GOTO1770
1280 IF(IZ#P1$)THENIF(P1(SV)=C)THENP1(SV)=SCELSEGOTO1400
1290 IF(IZ#P2$)THENIF(P2(SV)=C)THENP2(SV)=SCELSEGOTO1400
1300 GG=1:HH=6:WP=0:IF(Z#P1$)THENSU(1)=0ELSESU(2)=0
1310 GOSUB1490
1320 GG=7:HH=13:WP=1:IF(Z#P1$)THENTU(1)=0ELSETU(2)=0
1330 GOSUB1490
1340 IF(IN3$)THENN=3:JZ=0ELSEIF(IN<6ANDN>3)THENN=6:JZ=0
1350 J$="":H$="":Q=0:S(W)=0:FORW=1TO5:S(W)=0:NEXTW
1360 PRINT@338,E$;PRINT@260,Q$;E=USR(BV+20)
1370 RES=INKEY$:IFRES#""THEN1370
1380 PRINT@337,S$;QE=QE+1:IF(QE=26)THEN1720
1390 GOTO090
1400 PRINT@337,M$;E=USR(BW):PRINT@337,S$;GOTO1220
1410 FORJ=1TOX:NEXTJ:X=0:RETURN
1420 'OUT 254,16 REM SYSTEM 00 EXT CASSETTE
1430 FORA=TEXTTOTEXT+28:READR$POKEA,B:NEXT
1440 ONERRORGOTO1450:DEFUSR$=TEXT:RETURN
1450 POKE 16526,TEXT AND 255 : POKE 16527, INT(TEXT/256) : RETURN
1460 DATA205,127,105,229,193,165,16,254,58,61,64,246,2,203
1470 DATA215,211,255,65,16,254,230,253,211,255,193,16,233,201
1480 NZ=1600+RND(55):E=USR(NZ):GOSUB1710:RETURN
1490 FORPQ=GGTOHH:IF(WP=0)THENPR=403+PQ*6ELSEPR=427+((PQ-6)*64)
1500 IF(IZ#P1$)THEN1540
1510 IF(P2(PQ)=C)THENQ0=0:GOTO1520ELSEPRINT@PR,USINGU$;P2(PQ);:Q
B=P2(PQ)
1520 IF(WP=0)THENSU(2)=SU(2)+Q0ELSETU(2)=TU(2)+Q0
1530 GOTO1560
1540 IF(P1(PQ)=C)THENQ0=0:GOTO1550ELSEPRINT@PR,USINGU$;P1(PQ);:Q
B=P1(PQ)
1550 IF(WP=0)THENSU(1)=SU(1)+Q0ELSETU(1)=TU(1)+Q0
1560 NEXTPQ
1570 IF(IZ#P1$)THENQ4=1ELSEQ4=2
1580 IF(WP)THEN1620ELSEPT(Q4)=0
1590 IF(SU(Q4)>63)THENBS(Q4)=35ELSEBS(Q4)=0
1600 PT(Q4)=SU(Q4)+BS(Q4)
1610 PRINT@851,USINGU$;BS(Q4);:PRINT@913,USINGV$;PT(Q4);:RETURN
1620 PRINT@937,USINGV$;TU(Q4);:TS(Q4)=PT(Q4)+TU(Q4)
1630 PRINT@885,USINGV$;TS(Q4);:RETURN
1640 FOREE=1TO5:R(EE)=S(EE):NEXTEE
1650 M=0
1660 FORW=1TO4:IF(R(W)<R(W+1))THEN1680
1670 W1=R(W):R(W)=R(W+1):R(W+1)=W1:M=M+1
1680 NEXTW
1690 IF(M>0)THEN1650
1700 FORW=1TO5:PRINT@752+W*2,R(W);:NEXTW
1710 RETURN
1720 PRINT@275,"==== Game Finished ==="
1730 IF(TS(1)>TS(2))THENPRINT@328,P1$;P$;TS(1);X$;TS(2);0$;P2$;
1740 IF(TS(2)>TS(1))THENPRINT@328,P2$;P$;TS(2);X$;TS(1);0$;P1$;
1750 IF(TS(1)=TS(2))THENPRINT@330,P$;TS(1);PX$;TS(2);P0$;
1760 RES=INKEY$:IFRES#""THEN1760ELSERUN
1770 PRINT@337,S$;:YG=0:GOTO1210
1780 GOSUB1430

```

```

330 CHAIN "MODULE1/BAS",ALL
340 CHAIN "MODULE3/BAS",ALL
350 CHAIN "MODULE4/BAS",ALL
360 CHAIN "MODULE6/BAS",ALL
370 CHAIN "MODULE7/BAS",ALL
380 CHAIN "MODULE8/BAS",ALL
390 CHAIN "MODULE5/BAS",ALL
400 CHAIN "MODULE2/BAS",ALL
410 REM WRITTEN BY
SUNBURST SOFTWARE SERVICES

10 REM
20 COMMON Z0$,Z1$,Z2$,Z3$,Z4$,Z5$,Z6$,Z7$,A$(1),LB(1),DT$,DE$,RF$,
PR$,DB$,CR$,AD$,VU,VW
30 GOTO 410
40 AD$ = ""
50 FOR T = 1 TO LN
60 GOSUB 130 : IF IN$ = CHR$(13) THEN 110 ELSE IF IN$ = CHR$(8)
THEN 90 ELSE IF IN$ = CHR$(32) THEN GOSUB 150
70 AD$ = AD$ + IN$ : PRINT @PA,AD$ : NEXT : RETURN
80 NEXT : RETURN
90 IF T < = 1 THEN 60 ELSE T = T - 1
100 AD$ = LEFT$(AD$, LEN(AD$) - 1) : PRINT @PA,AD$ : ** : GOTO
60
110 IF FL = 0 THEN BL$ = STRING$(LN - LEN(AD$), " ") : AD$ = AD$
+ BL$ : PRINT @PA,AD$ : RETURN
120 BL$ = STRING$(LN - LEN(AD$), "0") : AD$ = AD$ + BL$ : PRINT
@PA,AD$ : RETURN
130 IN$ = "" : IN$ = INKEY$ : GOSUB 140 : IF IN$ = "" THEN 130 E
LSE RETURN
140 PRINT @PA,AD$;CHR$(143);: RETURN
150 IF FL = 0 THEN RETURN
160 IN$ = "0" : RETURN
170 T = 1
180 IF T > 5 THEN AD$ = "0" : RETURN
190 IF MID$(AD$,T,1) = "0" THEN T = T + 1 : GOTO 180
200 AD$ = STRING$(T - 1,32) + RIGHT$(AD$,6 - T) : RETURN
210 PRINT @160, CHR$(31) : RETURN
220 CHAIN "MODULE0/BAS",250,ALL
230 PRINT @190,"KEYBOARD INPUT"
240 PRINT @320,Z7$;" **/**/** : PRINT @400,"REF NO. ****";
: PRINT @480,"DETAILS *****"; : PRINT
@560,"PREFIX ** : PRINT @640,"ACC NO. ****"; : PRINT @720,Z6$;
" ****-**";
250 PRINT @240,"RECORD NO. ";I;
260 PRINT @800,Z5$;" ****-**"; : PRINT @880,"CORRECT (Y/N) **;
: RETURN
270 GOSUB 210 : GOSUB 230
280 FL = 1 : PA = 328 : LN = 2 : GOSUB 40 : DT$ = AD$ : IF DT$ =
"00" THEN RETURN
290 GOSUB 300 : GOSUB 310 : GOSUB 320 : GOSUB 330 : GOSUB 340 :
FL = 1 : GOSUB 350 : GOSUB 360 : GOSUB 370 : GOSUB 380 : FL = 0
: GOTO 390
300 PA = 331 : GOSUB 40 : DT$ = DT$ + AD$ : PA = 334 : GOSUB 40
: DT$ = DT$ + AD$ : RETURN
310 PA = 408 : LN = 4 : FL = 0 : GOSUB 40 : RF$ = AD$ : RETURN
320 PA = 488 : LN = 31 : GOSUB 40 : DE$ = AD$ : RETURN
330 PA = 568 : LN = 2 : GOSUB 40 : PR$ = AD$ : RETURN
340 PA = 648 : LN = 3 : GOSUB 40 : PR$ = PR$ + AD$ : RETURN

```

\*\*\*\* VER 3.0 HOUSEHOLD ACCOUNTING \*\*\*\*

## MODEL 4

```

10 CLEAR 50 : VV = INT((MEM - 2000) / 69) : VV = - VV * (VV < 6
98) - (VV > 697) * 697:REM
MODULE0/BAS

```

THE LARGEST MODULE, (MODULE6/BAS), IS 4203 BYTES LONG

```

20 CLEAR 62 * VV + 700 : VV = INT((FRE(Z0$) - 700) / 62) : DIM
A$(VV + 1),LB(VV + 1) : F3$ = " #####.##" : F4$ = " #####.###.
##-" : F5$ = " " + F3$ + "-"
30 W = 1 : S0$ = "Press any key to continue * : Z0$ = "Total" :
Z1$ = "Memory" : Z2$ = "Account" : Z3$ = "Select Function * :
Z4$ = "Journal" : Z5$ = "Credit" : Z6$ = "Debit" : Z7$ = "Date"
40 PRINT CHR$(15)
50 GOSUB 230 : PRINT @400,"Maximum number of records = ";VV : PR
INT @800,S0$ : PRINT@1840,"..... Written by Sunb
ur Software Services ....."; PA = 826 : LN = 1
: GOSUB 60 : GOTO 250
60 AD$ = ""
70 FOR T = 1 TO LN
80 GOSUB 150 : IF IN$ = CHR$(13) THEN 130 ELSE IF IN$ = CHR$(8)
THEN 110 ELSE IF IN$ = CHR$(32) THEN GOSUB 170
90 AD$ = AD$ + IN$ : PRINT @PA,AD$ : NEXT : RETURN
100 NEXT : RETURN
110 IF T < = 1 THEN 80 ELSE T = T - 1
120 AD$ = LEFT$(AD$, LEN(AD$) - 1) : PRINT @PA,AD$ : ** : GOTO
80
130 IF FL = 0 THEN BL$ = STRING$(LN - LEN(AD$), " ") : AD$ = AD$
+ BL$ : PRINT @PA,AD$ : RETURN
140 BL$ = STRING$(LN - LEN(AD$), "0") : AD$ = AD$ + BL$ : PRINT
@PA,AD$ : RETURN
150 IN$ = "" : IN$ = INKEY$ : GOSUB 160 : IF IN$ = "" THEN 150 E
LSE RETURN
160 PRINT @PA,AD$;CHR$(143);: RETURN
170 IF FL = 0 THEN RETURN
180 IN$ = "0" : RETURN
190 T = 1
200 IF T > 5 THEN AD$ = "0" : RETURN
210 IF MID$(AD$,T,1) = "0" THEN T = T + 1 : GOTO 200
220 AD$ = STRING$(T - 1,32) + RIGHT$(AD$,6 - T) : RETURN
230 CLS : PRINT @0,"**
Household Accounting
Ver
5.0 for the Model 4
** : PRINT @80,"***
C) 1st November 1983 M i c r o - 8 0 P t y L t d
***";
240 PRINT @160, CHR$(31) : RETURN
250 COMMON VV,A$(1),LB(1),F3$,F4$,F5$,W,S0$,Z0$,Z1$,Z2$,Z3$,Z4$,Z5
$,Z6$,Z7$,PA,LN,AD$,DT$,RF$,DE$,PR$,DB$,CR$,P
260 P=0:GOSUB 230:PRINT@196,"MENU":PRINT:PRINT*1 = Keyboar
d Input
: 5 = Save Data"
270 PRINT:PRINT*2 = Load Data
: 6 =
Print Journals"
280 PRINT:PRINT*3 = Read Memory
: 7 =
Lineprinter Utility"
290 PRINT:PRINT*4 = Edit Memory
: 8 =
Ledger Accounts"
300 PRINT @1200,Z3$ : PA = 1216 : LN = 1 : GOSUB 60
310 AD = VAL(AD$) : IF AD < 1 OR AD > 8 THEN 300
320 ON AD GOTO 330,400,340,350,360,370,380

```

```

340 FL = 0 : GOSUB 220 : PRINT @428,"ENTER FILENAME *****":
PA = 443 : LN = 8 : GOSUB 40 : NM$ = AD$ : RETURN
350 PRINT @492,"PRESS ANY KEY WHEN DEVICE READY OR (E)SCAPE *":
: PA = 538 : LN = 1 : GOSUB 40 : RETURN
360 CHAIN "MODULE0/BAS",250,ALL

10 REM
20 GOTO 220
30 AD$ = ""
40 FOR T = 1 TO LN
50 GOSUB 120 : IF IN$ = CHR$(13) THEN 100 ELSE IF IN$ = CHR$(8)
THEN 80 ELSE IF IN$ = CHR$(32) THEN GOSUB 140
60 AD$ = AD$ + IN$ : PRINT @PA,AD$: NEXT : RETURN
70 NEXT : RETURN
80 IF T < 1 THEN 50 ELSE T = T - 1
90 AD$ = LEFT$(AD$, LEN(AD$) - 1) : PRINT @PA,AD$:**": GOTO 5
0
100 IF FL = 0 THEN BL$ = STRING$(LN - LEN(AD$), " ") : AD$ = AD$
+ BL$ : PRINT @PA,AD$: RETURN
110 BL$ = STRING$(LN - LEN(AD$), "0") : AD$ = AD$ + BL$ : PRINT
@PA,AD$: RETURN
120 IN$ = "" : IN$ = INKEY$ : GOSUB 130 : IF IN$ = "" THEN 120 E
LSE RETURN
130 PRINT @PA,AD$:CHR$(143): RETURN
140 IF FL = 0 THEN RETURN
150 IN$ = "0" : RETURN
160 T = 1
170 IF T > 5 THEN AD$ = " 0" : RETURN
180 IF MID$(AD$,T,1) = "0" THEN T = T + 1 : GOTO 170
190 AD$ = STRING$(T - 1,32) + RIGHT$(AD$,6 - T) : RETURN
200 CLS : PRINT @0,**"
Household Accounting Ver
5.0 for the Model 4
***": PRINT @80,**"
C) 1st November 1983 M i c r o - 8 0 P t y L t d
***"

210 PRINT @160, CHR$(31): RETURN
220 COMMON VV,A$(1),LB(1),F3$,F4$,F5$,W,S0$,Z0$,Z1$,Z2$,Z3$,Z4$,Z5
$,Z6$,Z7$,PA,LN,AD$,DT$,RF$,DE$,PR$,DB$,CR$,P
230 GOSUB 210 : GOSUB 240 : GOSUB 280 : GOTO 340
240 PRINT @190,"Contents of "Z1$:
250 PRINT @240,Z7$: REF DETAILS
CC NO "Z6$:
260 IF P THEN LPRINT Z7$: REF DETAILS
ACC NO "Z6$:
270 RETURN
280 L = 1 : FOR I = 1 TO W - 1
290 GOSUB 330 : GOSUB 350 : PRINT VX$
300 L = L + 1 : IF L = 20 THEN L = 1 : PRINT @1840,S0$: PA = 1
866 : LN = 1 : GOSUB 30 : GOSUB 210 : GOSUB 240
310 NEXT I
320 PRINT @1840,"END OF DATA - "ISO$: PA = 1880 : LN = 1 : GOS
UB 30 : RETURN
330 V1$ = LEFT$(A$(1),2) : V2$ = MID$(A$(1),3,2) : V3$ = MID$(A$
(1),5,2) : V4$ = MID$(A$(1),7,4) : V5$ = MID$(A$(1),11,31) : V6$
= MID$(A$(1),42,2) : V7$ = MID$(A$(1),44,3) : V8$ = MID$(A$(1),
47,8) : V9$ = MID$(A$(1),55,8) : RETURN
340 CHAIN "MODULE0/BAS",250,ALL
350 VX$ = V1$ + "/" + V2$ + "/" + V3$ + " " + V4$ + " " + V5$ +
" " + V6$ + V7$ + " " + V8$ + " " + V9$ : RETURN

```



```

420 PRINT @328,V1$; : PRINT @331,V2$; : PRINT @334,V3$; : PRINT
@408,V4$; : PRINT @488,V5$; : PRINT @568,V6$; : PRINT @648,V7$;
: PRINT @727,V8$; : PRINT @807,V9$;
430 PA = 967 : LN = 1 : GOSUB 30 : IF AD$ = "I" THEN PRINT @960,
"ADVANCE"; : I = I + 1 : IF I > VV THEN I = VV
440 IF AD$ = "+" THEN PRINT @960,"ADVANCE"; : I = I + 1 : IF I
> VV THEN I = VV
450 IF AD$ = "-" THEN I = I - 1 : PRINT @960,"REVERSE"; : IF I <
= 0 THEN I = 1
460 IF AD$ = "=" THEN I = I - 10 : PRINT @960,"REVERSE"; : IF I
< = 0 THEN I = 1
470 IF AD$ = " " THEN 650
480 IF AD$ = "E" THEN PRINT @960,"* EDIT *"; : GOSUB 500 : GOTO
380
490 GOTO 390
500 FL = 1 : PA = 328 : LN = 2 : GOSUB 30 : IF AD$ = "00" THEN D
T$ = V1$ + V2$ + V3$ : PRINT @328,V1$;"/";V2$;"/";V3$; : GOTO 52
0
510 DT$ = AD$ : GOSUB 260
520 GOSUB 270 : IF RF$ = " " THEN RF$ = V4$ : PRINT @408,RF$;
530 GOSUB 280 : IF DE$ = " " THEN DE$ = V5$ : PRINT @488,DE$;
540 GOSUB 290 : IF PR$ = " " THEN P1$ = V6$ : PRINT @568,V6$; :
GOTO 560
550 P1$ = AD$
560 GOSUB 300 : IF AD$ = " " THEN PR$ = P1$ + V7$ : PRINT @648
,V7$; : GOTO 580
570 PR$ = P1$ + AD$
580 FL = 0 : GOSUB 310 : IF AD$ = " " THEN DB$ = V8$ : PRINT
@727,V8$; : GOTO 600
590 GOSUB 160 : GOSUB 320
600 FL = 0 : GOSUB 330 : IF AD$ = " " THEN CR$ = V9$ : PRINT
@807,V9$; : GOTO 620
610 GOSUB 160 : GOSUB 340
620 GOSUB 350
630 FL = 0 : RETURN
640 V1$ = LEFT$(A$(I),2) : V2$ = MID$(A$(I),3,2) : V3$ = MID$(A$
(I),5,2) : V4$ = MID$(A$(I),7,4) : V5$ = MID$(A$(I),11,31) : V6$
= MID$(A$(I),42,2) : V7$ = MID$(A$(I),44,3) : V8$ = MID$(A$(I),
47,8) : V9$ = MID$(A$(I),55,8) : RETURN
650 CHAIN "MODULE0/BAS",250,ALL

10 REM
MODULE5/BAS
20 COMMON VV,A$(1),LB(1),F3$,F4$,F5$,W,S0$,Z0$,Z1$,Z2$,Z3$,Z4$,Z5$
,Z6$,Z7$,PA,LN,AD,AD$,DT$,RF$,DE$,PR$,DB$,CR$,P
30 GOTO 230
40 AD$ = ""
50 FOR T = 1 TO LN
60 GOSUB 130 : IF IN$ = CHR$(13) THEN 110 ELSE IF IN$ = CHR$(8)
THEN 90 ELSE IF IN$ = CHR$(32) THEN GOSUB 150
70 AD$ = AD$ + IN$ : PRINT @PA,AD$; : NEXT : RETURN
80 NEXT : RETURN
90 IF T < = 1 THEN 60 ELSE T = T - 1
100 AD$ = LEFT$(AD$, LEN(AD$) - 1) : PRINT @PA,AD$;";"; : GOTO 5
0
110 IF FL = 0 THEN BL$ = STRING$(LN - LEN(AD$), " ") : AD$ = AD$
+ BL$ : PRINT @PA,AD$; : RETURN
120 BL$ = STRING$(LN - LEN(AD$), "0") : AD$ = AD$ + BL$ : PRINT
@PA,AD$; : RETURN
130 IN$ = "" : IN$ = INKEY$ : GOSUB 130 : IF IN$ = "" THEN 120 E
LSE RETURN
140 PRINT @PA,AD$;CHR$(143); : RETURN
150 IF FL = 0 THEN RETURN
160 IN$ = "0" : RETURN
170 T = 1
180 IF T > 5 THEN AD$ = " " : RETURN
190 IF MID$(AD$,T,1) = "0" THEN T = T + 1 : GOTO 170
200 AD$ = STRING$(T - 1,32) + RIGHT$(AD$,6 - T) : RETURN
210 CLS : PRINT @0,"***
Household Accounting Ver
5.0 for the Model 4
C) 1st November 1983 M i c r o - 8 0 P t y L t d
***";
220 PRINT @160, CHR$(31); : RETURN
230 COMMON VV,A$(1),LB(1),F3$,F4$,F5$,W,S0$,Z0$,Z1$,Z2$,Z3$,Z4$,Z5$
,Z6$,Z7$,PA,LN,AD,AD$,DT$,RF$,DE$,PR$,DB$,CR$,P;GOTO 360
240 PRINT @320,Z7$; " ***/****"; : PRINT @400,"REF NO. ****";
: PRINT @480,"DETAILS *****"; : PRINT
@560,"PREFIX **"; : PRINT @640,"ACC NO. ****"; : PRINT @720,Z6$;
" ****";
250 PRINT @240,"RECORD NO. ";I;
260 PRINT @200,Z5$; " ****"; : RETURN
270 PA = 331 : GOSUB 30 : DT$ = DT$ + AD$ : PA = 334 : GOSUB 30
: DT$ = DT$ + AD$ : RETURN
280 PA = 488 : LN = 4 : FL = 0 : GOSUB 30 : RF$ = AD$ : RETURN
290 PA = 568 : LN = 31 : GOSUB 30 : DE$ = AD$ : RETURN
300 PA = 648 : LN = 2 : GOSUB 30 : PR$ = AD$ : RETURN
310 PA = 727 : LN = 3 : GOSUB 30 : PR$ = PR$ + AD$ : RETURN
320 PA = 727 : LN = 5 : GOSUB 30 : GOSUB 160 : DB$ = AD$ + " " :
RETURN
330 PA = 733 : LN = 2 : GOSUB 30 : DB$ = DB$ + AD$ : RETURN
340 PA = 807 : LN = 5 : GOSUB 30 : GOSUB 160 : CR$ = AD$ + " " :
RETURN
350 PA = 813 : LN = 2 : GOSUB 30 : CR$ = CR$ + AD$ : FL = 0 : RE
TURN
360 A$(1) = DT$ + RF$ + DE$ + PR$ + DB$ + CR$ : RETURN
370 I = 1 : FL = 0
380 PRINT @213,"EDIT ";Z1$;
390 GOSUB 210 : GOSUB 230 : PRINT @832,"
"; : PRIN
T @960,"SELECT *";
400 PRINT @240,"RECORD NO. ";I;
410 IF V1$ = "" THEN GOSUB 230 : PRINT @832,"
";

```

```

200 CLS : PRINT @0, "***
5.0 for the Model 4
C) 1st November 1983 M i c r o - 8 0 P t y L t d
***;
Household Accounting Ver
***; : PRINT @0, ***
210 PRINT @160, CHR$(31); : RETURN
220 COMMON VV, A$(1), LB(1), F3$, F4$, F5$, W, S0$, Z0$, Z1$, Z2$, Z3$, Z4$, Z5
$, Z6$, Z7$, PA, LN, AD, AD$, DT$, RF$, DE$, PR$, DB$, CR$, P: GOTO 260
230 PRINT @320, Z7$; " REF DETAILS
CC NO "Z6$; "Z5$
240 IF P THEN LPRINT Z7$; " REF DETAILS
ACC NO "Z6$; "Z5$
250 RETURN
260 GOSUB 200 : PRINT @190, Z4$; "s Available"; PRINT: PRINT "1 = Pri
nt Ledger Balances : 4 = GJ General "Z4$; PRINT "2 =
CP Cash Payments "Z4$; " : 5 = SJ Sales "Z4$; PRINT "3
= CR Cash Received "Z4$; " :
270 PRINT " 6 = Return To Main Menu"
280 PRINT @832, Z3$; : PA = 848 : LN = 1 : GOSUB 30 : AD = VAL(AD
$)
290 IF AD < 1 OR AD > 6 THEN 280
300 ON AD GOTO 560, 310, 320, 330, 340, 530
310 PT$ = "CASH PAYMENTS" : KA$ = "CP" : GOTO 350
320 PT$ = "CASH RECEIVED" : KA$ = "CR" : GOTO 350
330 PT$ = "GENERAL" : KA$ = "GJ" : GOTO 350
340 PT$ = "SALES" : KA$ = "SJ"
350 GOSUB 360 : GOSUB 390 : GOTO 410
360 PRINT @832, "IS THE PRINTER REQUIRED (Y/N) *"; : PA = 862 : L
N = 1 : GOSUB 30
370 IF AD$ < "N" AND AD$ < "Y" THEN 360
380 P = (AD$ = "Y") : RETURN
390 ZK$ = " : PRINT @896, "WHICH "Z7$; " DO YOU REQUIRE **/**/*
"; FL = 1 : PA = 922 : LN = 2 : GOSUB 30 : E$ = AD$ : IF E$ =
"00" THEN E$ = "S" : ZK$ = "ALL" : GOTO 410
400 PA = 925 : GOSUB 30 : E$ = E$ + AD$ : PA = 928 : GOSUB 30 :
E$ = E$ + AD$ : RETURN
410 GOSUB 210 : PRINT : PRINT PT$; " "Z4$; " FOR "ZK$; " ("Z7$; "
"; E$ : IF P THEN LPRINT : LPRINT PT$; " "Z4$; " FOR "ZK$; " ("Z7$;
"); E$
420 DT$ = 0 : CT# = 0 : BL# = 0 : GOSUB 230
430 FOR I = 1 TO W
440 IF MID$(A$(I), 42, 2) < > KA$ THEN 490
450 IF E$ = "S" THEN 470
460 IF E$ < > LEFT$(A$(I), 6) THEN 490
470 GOSUB 540 : GOSUB 550 : PRINT VX$ : IF P THEN LPRINT VX$
480 DR# = VAL( MID$(A$(I), 47, 8)) : CR# = VAL( MID$(A$(I), 55, 8))
: DT# = DT# + DR# : CT# = CT# + CR# : BL# = BL# + DR# - CR#
490 NEXT
500 PRINT : PRINT Z0$; TAB(36); : PRINT USING F3$; DT#; CT# : PRIN
T "BALANCE "; : PRINT USING F4$; BL#
510 IF P THEN LPRINT : LPRINT Z0$; TAB(36); : LPRINT USING F3$; D
T#; CT# : LPRINT "BALANCE "; : LPRINT USING F4$; BL#
520 PRINT @960, "PRINTOUT COMPLETE - "F50$; : PA = 1006 : LN = 1
: GOSUB 30 : GOTO 260
530 CHAIN "MODULE0/BAS", 250, ALL
540 V1$ = LEFT$(A$(I), 2) : V2$ = MID$(A$(I), 3, 2) : V3$ = MID$(A$(
(I), 5, 2) : V4$ = MID$(A$(I), 7, 4) : V5$ = MID$(A$(I), 11, 3) : V6$
= MID$(A$(I), 42, 2) : V7$ = MID$(A$(I), 44, 3) : V8$ = MID$(A$(I),
47, 8) : V9$ = MID$(A$(I), 55, 8) : RETURN
550 VX$ = V1$ + " / " + V2$ + " / " + V3$ + " " + V4$ + " " + V5$ +
" " + V6$ + V7$ + " " + V8$ + " " + V9$ : RETURN

```

```

130 IN$ = " : IN$ = INKEY$ : GOSUB 140 : IF IN$ = " THEN 130 E
LSE RETURN
140 PRINT @PA, AD$; CHR$(143); : RETURN
150 IF FL = 0 THEN RETURN
160 IN$ = "0" : RETURN
170 T = 1
180 IF T > 5 THEN AD$ = " 0" : RETURN
190 IF MID$(AD$, T, 1) = "0" THEN T = T + 1 : GOTO 180
200 AD$ = STRING$(T - 1, 32) + RIGHT$(AD$, 6 - T) : RETURN
210 CLS : PRINT @0, ***
Household Accounting Ver
5.0 for the Model 4
***; : PRINT @0, ***
C) 1st November 1983 M i c r o - 8 0 P t y L t d
***;
220 PRINT @160, CHR$(31); : RETURN
230 X1$ = "SAVE TO" : GOSUB 310
240 GOSUB 330
250 IF SF = 2 THEN 360
260 GOSUB 340
270 GOSUB 350 : IF AD$ = "E" THEN 230
280 IF SF = 1 THEN OPEN "0", 1, NM$
290 IF SF = 1 THEN PRINT #1, W : FOR I = 1 TO W : PRINT #1, A$(I)
: NEXT : CLOSE
300 GOTO 230
310 GOSUB 220 : PRINT @190, "DATA "; LEFT$(X1$, 4); "
320 PRINT:PRINT
330 PRINT "2 = EXIT TO MENU" : RETURN
340 IF SF < 1 OR SF > 3 THEN 330 ELSE RETURN
340 FL = 0 : GOSUB 220 : PRINT @428, "ENTER FILENAME *****"; :
PA = 443 : LN = 8 : GOSUB 40 : NM$ = AD$ : RETURN
350 PRINT @492, "PRESS ANY KEY WHEN DEVICE READY OR (E)SCAPE *
"; PA = 538 : LN = 1 : GOSUB 40 : RETURN
360 CHAIN "MODULE0/BAS", 250, ALL
MODULE6/BAS
10 REM
20 GOTO 220
30 AD$ = "
40 FOR T = 1 TO LN
50 GOSUB 120 : IF IN$ = CHR$(13) THEN 100 ELSE IF IN$ = CHR$(8)
THEN 80 ELSE IF IN$ = CHR$(32) THEN GOSUB 140
60 AD$ = AD$ + IN$ : PRINT @PA, AD$; : NEXT : RETURN
70 NEXT : RETURN
80 IF T < 1 THEN 50 ELSE T = T - 1
90 AD$ = LEFT$(AD$, LEN(AD$) - 1) : PRINT @PA, AD$; ***; : GOTO 5
0
100 IF FL = 0 THEN BL$ = STRING$(LN - LEN(AD$), " ") : AD$ = AD$
+ BL$ : PRINT @PA, AD$; : RETURN
110 BL$ = STRING$(LN - LEN(AD$), "0") : AD$ = AD$ + BL$ : PRINT
@PA, AD$; : RETURN
120 IN$ = " : IN$ = INKEY$ : GOSUB 130 : IF IN$ = " THEN 120 E
LSE RETURN
130 PRINT @PA, AD$; CHR$(143); : RETURN
140 IF FL = 0 THEN RETURN
150 IN$ = "0" : RETURN
160 T = 1
170 IF T > 5 THEN AD$ = " 0" : RETURN
180 IF MID$(AD$, T, 1) = "0" THEN T = T + 1 : GOTO 170
190 AD$ = STRING$(T - 1, 32) + RIGHT$(AD$, 6 - T) : RETURN

```

```

560 GOSUB 200 : GOSUB 360 : GOSUB 200 : GOSUB 710 : PRINT @463, "
  * * * W A I T * * * : FOR I = 1 TO W - 1 : GOSUB 540 : LB(I)
  = VAL ( RIGHT$(V7$,3) ) : NEXT I : J = 0 : DT# = 0 : CT# = 0 : BL#
  = 0
570 GOSUB 210 : PRINT : PRINT "ACC NO.": TAB(16);Z6$;"S": TAB(34
);Z5$;"S": TAB(55);Z0$
580 IF P THEN LPRINT : LPRINT "ACC NO.": TAB(16);Z6$;"S": TAB(34
);Z5$;"S": TAB(55);Z0$
590 TT# = 0 : DR# = 0 : CR# = 0 : J = J + 1
600 VF = LB(J) : FOR I = 1 TO W : IF LB(I) = VF AND LB(I) < > 0
THEN GOSUB 700
610 NEXT I : LB(J) = 0 : IF J = W THEN 670 ELSE 620
620 IF VF = 0 THEN 590
630 TT# = DR# - CR# : DT# = DT# + DR# : CT# = CT# + CR# : BL# =
  BL# + TT#
640 PRINT " : MID$(A$(J),44,3) : PRINT USING F5$;DR#;CR#;TT#
650 IF P THEN LPRINT " : MID$(A$(J),44,3) : LPRINT USING F5$;
  DR#;CR#;TT#
660 GOTO 590
670 PRINT : PRINT Z0$ : PRINT USING F5$;DT#;CT#;BL#
680 IF P THEN LPRINT : LPRINT Z0$ : LPRINT USING F5$;DT#;CT#;BL#
#
690 PRINT @1840,S0$ : PA = 1846 : LN = 1 : GOSUB 30 : GOTO 530
700 GOSUB 540 : DR# = DR# + VAL(V8$) : CR# = CR# + VAL(V9$) : LB
(1) = 0 : RETURN
710 PRINT @463,"* * * SORTING * * *" : FOR SC = 1 TO W - 1 : F
  OR SA = 1 TO W - 1
720 SA$ = MID$(A$(SA),28,3)
730 SB$ = MID$(A$(SA + 1),28,3) : IF SB$ = "" THEN GOTO 740 ELSE
  IF SA$ > SB$ THEN SB$ = A$(SA) : A$(SA) = A$(SA + 1) : A$(SA +
  1) = SB$
740 NEXT SA : NEXT SC : RETURN

10 REM
20 ON ERROR GOTO 100
30 GOTO 50
40 PRINT @160, CHR$(31) : RETURN
50 COMMON VV,A$(1),LB(1),F3$,F4$,F5$,W,S0$,Z0$,Z1$,Z2$,Z3$,Z4$,Z5$,
  Z6$,Z7$,PA,LN,AD,AD$,DT$,RF$,DE$,PR$,DB$,CR$,P
60 GOSUB 40 : PRINT @190,"LINEPRINTER UTILITY":PRINT:PRINT"TYPE
  HEADINGS OR NOTES AS REQUIRED":PRINT:PRINT"TYPE " : CHR$(34);"EXI
  T": CHR$(34);" TO RETURN TO MAIN MENU":PRINT CHR$(14)
70 M$ = "" : INPUT M$ : IF M$ = "EXIT" THEN PRINT CHR$(15):GOTO
  90
80 PRINT M$ : LPRINT M$ : GOTO 70
90 CHAIN "MODULE0/BAS",250,ALL
100 IF ERP = 57 THEN PRINT"Device I/O error":PRINT"PLEASE CONNEC
  T PRINTER OR TYPE EXIT":PRINT:RESUME 70

10 REM
20 GOTO 220
30 AD$ = ""
40 FOR T = 1 TO LN
50 GOSUB 120 : IF IN$ = CHR$(13) THEN 100 ELSE IF IN$ = CHR$(8)
  THEN 80 ELSE IF IN$ = CHR$(32) THEN GOSUB 140
60 AD$ = AD$ + IN$ : PRINT @PA,AD$ : NEXT : RETURN
70 NEXT : RETURN
80 IF T < = 1 THEN 50 ELSE T = T - 1

90 AD$ = LEFT$(AD$, LEN(AD$) - 1) : PRINT @PA,AD$;"* * * : GOTO 5
  + BL$ : PRINT @PA,AD$ : RETURN
110 BL$ = STRING$(LN - LEN(AD$),"0") : AD$ = AD$ + BL$ : PRINT
  @PA,AD$ : RETURN
120 IN$ = "" : IN$ = INKEY$ : GOSUB 130 : IF IN$ = "" THEN 120 E
  LSE RETURN
130 PRINT @PA,AD$;CHR$(143): RETURN
140 IF FL = 0 THEN RETURN
150 IN$ = "0" : RETURN
160 T = 1
170 IF T > 5 THEN AD$ = " 0" : RETURN
180 IF MID$(AD$,T,1) = "0" THEN T = T + 1 : GOTO 170
190 AD$ = STRING$(T - 1,32) + RIGHT$(AD$,6 - T) : RETURN
200 CLS : PRINT @0,***
  Household Accounting Ver
  5.0 for the Model 4
  C) 1st November 1983 M i c r o - 8 0 P t y L t d
  ***;
210 PRINT @160, CHR$(31) : RETURN
220 COMMON VV,A$(1),LB(1),F3$,F4$,F5$,W,S0$,Z0$,Z1$,Z2$,Z3$,Z4$,Z5$,
  Z6$,Z7$,PA,LN,AD,AD$,DT$,RF$,DE$,PR$,DB$,CR$,P:GOTO 290
230 PRINT @320,Z7$;" REF DETAILS
  CC NO "Z6$;" :Z5$
240 IF P THEN LPRINT Z7$;" REF DETAILS
  ACC NO "Z6$;" :Z5$
250 RETURN
260 PRINT @832,"IS THE PRINTER REQUIRED (Y/N) *": PA = 862 : L
  N = 1 : GOSUB 30
270 IF AD$ < > "N" AND AD$ < > "Y" THEN 260
280 P = (AD$ = "Y") : RETURN
290 GOSUB 210 : PRINT @190,"Ledger "Z2$;"s":PRINT:PRINT"Type " :
  CHR$(34);"999": CHR$(34);" To Exit"
300 GOSUB 260
310 PRINT @960,"Which "Z2$;" No. Do You Require ***": PA = 99
  3 : LN = 3 : FL = 1 : GOSUB 30 : N = VAL(AD$)
320 IF N < 1 OR N > 999 THEN 310
330 BL# = 0 : DT# = 0 : CT# = 0 : IF N = 999 THEN 480
340 GOSUB 200 : PRINT : PRINT Z2$;" NO. "AD$ : IF P THEN LPRIN
  T : LPRINT Z2$;" NO. "AD$
350 GOSUB 230 : FOR I = 1 TO W
360 IF N < > VAL ( MID$(A$(I),28,3) ) THEN 390
370 GOSUB 460 : GOSUB 470 : PRINT VX$ : IF P THEN LPRINT VX$
380 DR# = VAL ( MID$(A$(I),47,8) ) : CR# = VAL ( MID$(A$(I),55,8) )
  : DT# = DT# + DR# : CT# = CT# + CR# : BL# = BL# + DR# - CR#
390 NEXT
400 PRINT : PRINT Z0$; TAB(36) : PRINT USING F3$;DT#;CT#
410 IF P THEN LPRINT : LPRINT Z0$; TAB(36) : LPRINT USING F3$;D
  T#;CT#
420 PRINT Z2$;" BALANCE": PRINT USING F4$;BL#
430 IF P THEN LPRINT Z2$;" BALANCE": LPRINT USING F4$;BL#
440 PRINT : PRINT : IF P THEN LPRINT " : LPRINT "
450 GOTO 310
460 V1$ = LEFT$(A$(1),2) : V2$ = MID$(A$(1),3,2) : V3$ = MID$(A$
  (1),5,2) : V4$ = MID$(A$(1),7,4) : V5$ = MID$(A$(1),11,31) : V6$
  = MID$(A$(1),42,2) : V7$ = MID$(A$(1),44,3) : V8$ = MID$(A$(1),
  47,8) : V9$ = MID$(A$(1),55,8) : RETURN
470 VX$ = V1$ + "/" + V2$ + " + V3$ + " + V4$ + " + V5$ +
  " + V6$ + V7$ + " + V8$ + " + V9$ : RETURN
480 CHAIN "MODULE0/BAS",250,ALL

```



```

00010 *****
00020 *      WRITER 1.1      *
00030 *(C) 1983 G.D. WILLIAMSON*
00040 *****
00050 *
00060 *
00070 *This program is in two parts
00080 *
00090 *This is the loader
00100 *
00110 *
00120 SCRST    EQU    $BA      Basic's pointer to screen start
00130 ENDBAS  EQU    $1B      End of Basic program pointer
00140          ORG     0
00150 START   LDX     <ENDBAS  Get end of Basic pointer
00160          TFR     X,U      Keep it in U
00170          LEAX    $4CD,X   Length of our program
00180          STX     <ENDBAS  Redirect end of program pointer
00190          LEAX    REGIN,PCR Point to start of program
00200          LDY     #$4CD    Counter
00210 LOOP    LDA     ,X+      Get byte
00220          STA     ,U+      Reposition it
00230          LEAY    -1,Y     Counter down
00240          RNE     LOOP     ?Go again
00250          LDA     #$39     RTS code
00260          STA     START,PCR Do not allow to be exec'd again
00270          RTS            Back to Basic
00280 *
00290 *
00300 *This is the entry point for USR0
00310 *
00320 *
00330 BEGIN    PSHS     CC,X    Save them
00340          ORCC     #$50     Disable interrupts
00350          TST     <FLAG,PCR ?already set
00360          BNE     BACK1    If set, exit
00370          INC     <FLAG,PCR If not, set it
00380          LDX     $168     Get Basic's return address
00390          STX     1+RET1,PCR And save it
00400          LEAX    <CHROUT,PCR Point to our routine
00410 BACK     STX     $168     Direct Basic to our patch
00420 BACK1    ANDCC    #$AF    Enable interrupts
00430          PULS     X,CC,PC  Back to Basic
00440 *
00450 *
00460 *This is the entry point for USR1
00470 *
00480 *
00490          PSHS     X,CC     Save them
00500          ORCC     #$50     Disable interrupts
00510          TST     <FLAG,PCR ? set
00520          BEQ     BACK1    Back if so
00530          CLR     <FLAG,PCR If not, clear it
00540 RET1     LDX     #$1111   Dummy address - see line 390
00550          BRA     BACK     Reset RAM hook for Basic
00560 *
00570 *
00580 *Start of our main routine
00590 *
00600 *
00610 CHROUT    PSHS     U,D,X,Y,CC  Save all
00620          CMPA     #8           ?Backspace
00630          BNE     CRTRNG      If not look for more controls
00640          LDA     #$20         ASCII for space
00650          BSR     DSPLY       Go show it
00660          LEAX    -1,X        Back one pos
00670          BSR     SHOW        Rub out charac
00680          BRA     STOP        Exit
00690 CRTRNG    CMPA     #$20       ?Control charac
00700          RLO     STOP        Exit if so
00710          CMPA     #$7F        ?Graphics charac
00720          BHI     STOP        Exit if so
00730          RSR     DSPLY       Go show it
00740          BSR     SHOW
00750          BRA     STOP        Exit

```

```

00760 DSPLY  PSHS  A      Save charac
00770      LDD    <$88    Basic's cursor pos
00780      PSHS  B      Save LSB
00790      LSRA    MSB of A into carry bit
00800      RORB    And into MSBit of B
00810      LSRB    Shift into lower nybble
00820      LSRB
00830      LSRB
00840      LSRB
00850      LDA     #$0C    12 bytes/charac
00860      MUL     Modify position
00870      LDA     #$20    32 bytes/row
00880      MUL     Get row position
00890      TFR     D,X     Swap for later
00900      PULS    B      Retrieve it
00910      ANDB   #$1F    Clear bits 5-7
00920      ABX     Get column position
00930      LDD    <SCRST  Get screen start
00940      LFAX    D,X     Get position on screen
00950      PULS    A      Retrieve charac
00960      RTS
00970 SHOW   SUBA   #$20    Adjust for table
00980      LDB     #$0C    12 bytes/charac
00990      MUL     So modify
01000      LEAU    TABLE,PCR Point to start of table
01010      LEAU    D,U     Point to charac
01020      LDB     #$0C    12 bytes/charac
01030 DSPLY1 LDA     ,U+    Get byte from table
01040      STA     ,X     Display it
01050      LEAX    $20,X    Down one row
01060      DECB    Counter down
01070      BNE     DSPLY1  ?Get more
01080      RTS        Back to sender
01090 STOP   PULS    U,D,X,Y,CC,PC Back to Basic
01100 *
01110 *
01120 FLAG   FCB     0      Store to show which USR routine is invoked
01130 *
01140 *
01150 *Start of character table
01160 TABLE FCB     0      SPACE
01170      FCB     0
01180      FCB     0
01190      FCB     0
01200      FCB     0
01210      FCB     0
01220      FCB     0
01230      FCB     0
01240      FCB     0
01250      FCB     0
01260      FCB     0
01270      FCB     0
01280      FCB     0      !
01290      FCB     0
01300      FCB     0
01310      FCB     0
01320      FCB     0
01330      FCB     0
01340      FCB     0
01350      FCB     0
01360      FCB     0
01370      FCB     0
01380      FCB     0
01390      FCB     0
01400      FCB     0      "
01410      FCB     $24
01420      FCB     $24
01430      FCB     0
01440      FCB     0
01450      FCB     0
01460      FCB     0
01470      FCB     0
01480      FCB     0
01490      FCB     0
01500      FCB     0

```

```

32C0: 0000 0041 6355 4941 4141 4141 0000 0241
32D0: 4141 5149 4543 4141 0000 007E 4141 417E 4040 4040
32F0: 0000 003E 4141 4141 4945 423D 0000 007E
3300: 4141 417E 4844 4241 0000 003E 4140 403E
3310: 0101 413E 0000 007F 0808 0808 0808 0808
3320: 0000 0041 4141 4141 4141 413E 0000 0041
3330: 4141 4141 4122 1408 0000 0041 4141 4141
3340: 4955 6341 0000 0041 4122 1408 0808 0000 007F
3350: 0000 0041 4141 2214 0808 0808 0000 007F
3360: 0102 0408 1020 407F 0000 003C 2020 2020
3370: 2020 203C 0000 0040 4020 1008 0402 0101
3380: 0000 003C 0404 0404 0404 043C 0000 0000
3390: 1C2A 4908 0808 0808 0808 0808 0808 1020
33A0: 7F20 1008 0000 0000 0000 0000 0000 0000
33B0: 0000 0000 0000 3C02 3E42 423C 0000 0040
33C0: 4040 7C42 4242 427C 0000 0000 0000 3C42
33D0: 4040 423C 0000 0002 0202 3E42 4242 423E
33E0: 0000 0000 0000 7C44 7C40 407C 0000 000A
33F0: 0808 081E 0808 0808 0808 0808 0808 3C42
3400: 4242 423E 0242 3C40 4040 7C42 4242 4242
3410: 0000 0000 1000 3010 1010 107C 0000 0000
3420: 0002 0006 0202 0242 3C40 4044 4870 1010
3430: 4844 4242 0000 0000 7749 4941 4141 0000 0000
3440: 0000 0000 5C62 4242 4242 0000 0000 3C42
3450: 4242 423C 0000 0000 0000 0000 7C42 427C
3460: 4040 4000 0000 3C42 4242 423E 0202 0300
3470: 0000 4C52 6040 4040 0000 0000 0000 3E40
3480: 3C02 027C 0000 0808 0808 0808 0808 0A04
3490: 0000 0000 0000 4242 4242 463A 0000 0000
34A0: 0000 4242 4242 2418 0000 0000 0000 4141
34B0: 4149 4977 0000 0000 0000 4224 1818 2442
34C0: 0000 0000 0000 4242 4242 423E 0242 3C00
34E0: 0000 7E02 0418 207E 0000 0000 0000 0000

```

\*\*\* TRACK RACER \*\*\*

HITACHI PEACH

```

10 REM TRACK RACER BY D.C. KELLY, 20 RUTH
ST. CORINDA, BRISBANE, 4075
20 WIDTH40: INPUT "DO YOU WANT DIRECTIONS
(Y/N) "; DR$: IF DR$="Y" THEN 220
30 RANDOMIZE (PEEK(&HFF00)*255+PEEK(&HFF00))
01: Z=20: B#=CHR$(254): D$=CHR$(94): S$=CHR
$(92): TIME$="00:00": SCREEN 0: WIDTH40
40 FOR Y=0 TO 24

```

```

12040 FCB 07E
12050 FCB 0
12060 FCB 0
12070 FCB 0
12080 FCB 0
12090 FCB 0
12100 FCB 0
12110 END START
3000: 9E1B 1F13 3069 04CD 9F1B 308D 0012 108E
3010: 04CD A680 A7C0 313F 26F8 8639 A78C E139
3020: 3411 1A50 6D8C 7F26 106C 8C7A BE01 68AF
3030: 8D00 1730 8C18 BF01 681C AF35 9134 111A
3040: 506D 8C62 27F3 6F8C 5D8E 1111 20E8 3477
3050: 8108 260A 8620 8D14 301F 8D30 2046 8120
3060: 2542 817F 223E 8D04 8D22 2038 3402 DC88
3070: 3404 4456 5454 5454 860C 3D86 203D 1F01
3080: 3504 C41F 3ADC BA30 8B35 0239 8020 C60C
3090: 3D33 8D00 1233 C8C6 0CA6 C0A7 8430 8820
30A0: 5A26 F639 35F7 0000 0000 0000 0000 0000
30B0: 0000 0000 0808 0808 0808 0808 0808 0000
30C0: 2424 7E24 2400 0008 3E49 281C 0A09 493E
30D0: 0800 0000 7051 7204 0810 2745 0700 000C
30E0: 1212 0C18 2542 4639 0000 0000 1810 0000
30F0: 0000 0000 0404 0404 0404 0408 0000 0000
3100: 0000 4122 147F 1422 4100 0000 0000 0008
3110: 0808 0808 0800 0000 0000 0000 0000 0018
3120: 0808 0808 0800 0000 0000 0000 0000 0000
3130: 1808 1000 0000 0000 7F00 0000 0000 0000
3140: 0000 0000 0000 1818 0000 0001 0102 0408
3150: 1020 4040 0000 003E 4143 4549 5161 413E
3160: 0000 0008 1808 0808 0808 083E 0000 003E
3170: 4101 0204 0810 207F 0000 003E 4101 010E
3180: 0101 413E 0000 0002 060A 1222 7F02 0202
3190: 0000 007F 4040 407E 0101 413E 0000 003E
31A0: 4140 407E 4141 413E 0000 007F 0101 0204
31B0: 0810 2020 0000 003E 4141 413E 4141 413E
31C0: 0000 003E 4141 413F 0101 413E 0000 0000
31D0: 0000 0018 1800 1018 0000 0000 0000 0018
31E0: 1800 1818 0810 0000 0204 0810 2010 0804
31F0: 0200 0000 0000 007F 007F 0000 0000 0000
3200: 2010 0804 0204 0810 2000 003E 4101 0204
3210: 0808 0808 0000 003E 4140 404E 5151 493E
3220: 0000 0008 1422 417F 4141 4141 0000 007E
3230: 2121 213E 2121 217E 0000 003E 4140 4040
3240: 4040 413E 0000 007E 2121 2121 2121 217E
3250: 0000 007F 4040 407E 4040 0000 007F 0000 007F
3260: 4040 407E 4040 4040 0000 007F 4040 4341
3270: 4141 417F 0000 0041 4141 417F 4141 4141
3280: 0000 001C 0808 0808 0808 081C 0000 0004
3290: 0404 0404 0404 441C 0000 0041 4244 4870
32A0: 4844 4241 0000 0040 4040 4040 4040 407F
32B0:

```

[illegible]



[illegible]

```

680 PRINT
690 DW=DW+1:IFDW=4THEN750
700 PRINT"The tally so far is:"PRINT"-----"
710 PRINT"Defender has won ";DW
720 PRINT"Challenger has won ";CW
730 PRINT:PRINT:INPUT"Press <ENTER> to start the next race";ZZ$:
GOTO120
740 FORG=1TO800:NEXTG:W1$=CH$:GOTO760
750 FORG=1TO800:NEXTG:W1$=DF$
760 CLS:PRINT"The Australia's cup has come to an end !:"
770 PRINT:PRINT"The winning yacht was ";W1$;" from ";
780 IFW1$=DF$THENPRINT"Australia !"ELSEPRINTCN$
790 PRINT
800 IFW1$=DF$THENPRINT"Australia has defended the cup yet again
!:"ELSEPRINTCN$;" has won the cup from Australia !:"
810 PRINT512;"
820 IFW1$=DF$THENPRINT"-- Australia's Cup!"ELSEPRINT"-- ";CN$;
""s Cup";
830 PRINT896,"";:INPUT"Press <ENTER> to see betting results";ZZ
$
840 IFW1$=DF$THENX$="D":XX$="d":XV=DVELSEX$="C":XX$="c":XV=CV
850 FORZ=1TONP:IFPB$(Z)=X$ORPB$(Z)=XX$THENPB(Z)=PB(Z)*XV:Z$(Z)="
won"ELSEPB(Z)=0:Z$(Z)="lost"
860 NEXTZ
870 CLS:PRINT"The betting results for all players are below.":PR
INTSTRING$(64,140);
880 FORZ=1TONP:PRINTPN$(Z);" has ";Z$(Z);
890 IFPB(Z)=0THENPRINT" !:"ELSEPRINT" ";PB(Z)
900 NEXTZ:PRINTSTRING$(64,140)
910 MS$="Press <ENTER> for another Australia's Cup":PP=960:SL=1:
GOSUB920:RUN
920 80$="":PRINTPP,MS$;" ";
930 FORT=1TOSL:PRINTCHR$(95);:NEXT:FORT=1TOSL:PRINTCHR$(24);:NEX
T
940 AS$=INKEY$
950 IFAS$=""THEN940
960 IFAS$=CHR$(8)ANDLEN(80$)=0THEN940
970 80$=80$+AS$:IFAS$=CHR$(13)THEN80$=LEFT$(80$,LEN(80$)-1):RETU
RN
980 PRINTAS$;:IFAS$=CHR$(8)THEN80$=LEFT$(80$,LEN(80$)-2):PRINTCH
R$(95);CHR$(24);:GOTO940
990 IFLEN(80$)=SLTHENRETURN
1000 GOTO940
1010 GOTO1350
1020 CLS:FORT=1TOSL:PRINT"
";T$(T):NEXT:FORT=1TOSL:PRINT
R$(13);
1030 PRINT8769,"You -->";:PRINT833,"are";:PRINT896,"here!";
1040 FORH=1TO25
1050 PRINT8780,"X";
1060 FORG=1TO900:NEXTG
1070 PRINT8780," ";
1080 FORG=1TO900:NEXTG
1090 IFINKEY$=""THENNEXT:GOTO1890ELSE1100
1100 CLS:MS$="Do you require instructions ?":PP=512:SL=1:GOSUB92
0:IF80$="N"OR80$="n"THENRETURN

```

```

190 IFCT>DTTHENPRINT80,"The Challenger ";CHR$(34);CH$;CHR$(34);"
leads. ";CHR$(30);
200 IFDT>CTTHENPRINT80,"The Defender ";CHR$(34);DF$;CHR$(34);" l
eads. ";CHR$(30);
210 GOTO120
220 GOTO220
230 IFDS=305THEN250ELSEDS=DS+DD
240 PRINT8DS,SR$;:RETURN
250 DC=DC+1:IFDC=1THENPRINT8DS,RR$;:RETURN
260 IFDC=2THENPRINT8DS,CS$;:RETURN
270 IFDC=3THENPRINT8DS,RL$;:RETURN
280 IFDC=4THENDC=0:PRINT8DS,SL$;:DD=-1:DL=DL+1
290 RETURN
300 IFDS<=260THEN320ELSEDS=DS+DD
310 PRINT8DS,SL$;:RETURN
320 DC=DC+1:IFDC=1THENPRINT8DS,RL$;:RETURN
330 IFDC=2THENPRINT8DS,CS$;:RETURN
340 IFDC=3THENPRINT8DS,RR$;:RETURN
350 IFDC=4THENDC=0:PRINT8DS,SR$;:DD=1:DL=DL+1
360 RETURN
370 DT=DT+1:IFDD=1THEN230ELSE300
380 CT=CT+1:IFCD=1THEN390ELSE460
390 IFCS=625THEN410ELSECS=CS+CD
400 PRINT8CS,SR$;:RETURN
410 CC=CC+1:IFCC=1THENPRINT8CS,RR$;:RETURN
420 IFCC=2THENPRINT8CS,CS$;:RETURN
430 IFCC=3THENPRINT8CS,RL$;:RETURN
440 IFCC=4THENDC=0:PRINT8CS,SL$;:CD=-1:CL=CL+1
450 RETURN
460 IFCS<=580THEN480ELSECS=CS+CD
470 PRINT8CS,SL$;:RETURN
480 CC=CC+1:IFCC=1THENPRINT8CS,RL$;:RETURN
490 IFCC=2THENPRINT8CS,CS$;:RETURN
500 IFCC=3THENPRINT8CS,RR$;:RETURN
510 IFCC=4THENDC=0:PRINT8CS,SR$;:CD=1:CL=CL+1
520 RETURN
530 FORG=1TO600:NEXTG:CLS:OUT254,0:PRINT"The Challenging ship ";
CH$;" from ";CN$;" has won!"
540 PRINT:PRINT
550 CT=(CT-DT)*6:MN=INT(CT/60):SE=CT-(MN*60)
560 PRINT"The Challenger won by ";MN;" minutes & ";SE;" seconds.
"
570 PRINT
580 CW=CW+1:IFCW=4THEN740
590 PRINT"The tally so far is:"PRINT"-----"
600 PRINT"Challenger has won ";CW
610 PRINT"Defender has won ";DW
620 PRINT:PRINT:INPUT"Press <ENTER> to start the next race";ZZ$:
GOTO120
630 FORG=1TO600:NEXTG:CLS:OUT254,0:PRINT"The Defending ship ";DF
$;" won!!"
640 PRINT:PRINT"Australia will be celebrating !:"
650 PRINT:PRINT
660 DT=(DT-CT)*6:MN=INT(DT/60):SE=DT-(MN*60)
670 PRINT"The Defender won by ";MN;" minutes & ";SE;" seconds.
"

```

```

1360 '* *
1370 '* *
1380 '* *
1390 '* *
1400 '* THESE LINES WILL CONTAIN THE GRAPHIC STRINGS *
1410 '* *
1420 '* *
1430 '* *
1440 '* *
1450 '* *
1460 '* *
1470 '* *
1480 '* *
1490 '* *
1500 '* *
1510 '* *
1520 '* *
1530 '* *
1540 '* *
1550 '* *
1560 '*****
1570 GOTO1020
1580 RESTORE:FORN=1TO1000:READS$:IFS$(<)"AUSSIE"THENNEXTI
1590 FORN=1TORND(10):READS$:NEXT:DF$=S$:FORT=1TO1000:READS$:IFS$
(<)"THEM"THENNEXTI
1600 U=RN(30):FORT=1TOU:READS$:NEXT:CH$=S$
1610 IFU<10THENCN$="America"
1620 IFU>10ANDU<20THENCN$="England"
1630 IFU>20ANDU<30THENCN$="New Zealand"
1640 X=RN(7):ONXGOTO1650,1660,1670,1680,1690,1700,1710
1650 D0$="2/1":DV=2:GOTO1720
1660 D0$="5/1":DV=5:GOTO1720
1670 D0$="10/1":DV=10:GOTO1720
1680 D0$="20/1":DV=20:GOTO1720
1690 D0$="40/1":DV=40:GOTO1720
1700 D0$="50/1":DV=50:GOTO1720
1710 D0$="100/1":DV=100:GOTO1720
1720 X=RN(7):ONXGOTO1730,1740,1750,1760,1770,1780,1790
1730 C0$="2/1":CV=2:GOTO1800
1740 C0$="5/1":CV=5:GOTO1800
1750 C0$="10/1":CV=10:GOTO1800
1760 C0$="20/1":CV=20:GOTO1800
1770 C0$="40/1":CV=40:GOTO1800
1780 C0$="50/1":CV=50:GOTO1800
1790 C0$="100/1":CV=100:GOTO1800
1800 RETURN
1810 IFPB(Z)>=0ANDPB(Z)<=100THENRETURN
1820 PP=PP-64:PRINT@960,"Sorry ";PN$(Z);", but you're bet is ill
egal!";PRINT@PP+64,CHR$(30);PB(Z)=0:Z=Z-1:RETURN
1830 RETURN
1840 DATA"AUSSIE",Australia III,Down Under,Kangaroo I,Kangaroo I
I,Koala I,Emu II,Aussie Crawl,Kookaburra I,Advance II,Wallaby I
1850 DATA"THEM",Yankee I,Yankee II,Liberty,Stars n' Stripes,Newp
ort,Yankee Doodle,Boston Strangler,Lost Angeles,Apple Pie,Titani
c I

```

```

1110 CLS:PRINT" Welcome to the ";CHR$(34);" A U S T
R A L I A ' S C U P";CHR$(34):PRINTSTRING$(64,140);
1120 PRINT"On the 26th of September 1983, the America's Cup was
lost by the American yacht 'Liberty' to the Australian yacht 'Aus
tralia II'."
1130 PRINT"This day will go down in history because a 132 year w
inning streak finally came to an end. The people of Australi
a rejoiced in Australia II's triumph and celebrated the win for
many days (and nights) after the glorious event."
1140 PRINT"The America's cup (affectionately known as The Auld M
ug) was transported to Australia and to the Royal Perth Yacht
Club where it was gazed upon in wonder by all."
1150 PRINT"The Auld Mug's previous home had been the New York Ya
cht Club which was renowned for its sly and devious tactics us
ed to stop the cup from leaving their shores.":PRINTSTRING$(64,1
40);
1160 PRINT@979,"Press any key to continue";
1170 IFPEEK(15359)=0THEN1170
1180 CLS:PRINT" The A U S T R A L I A ' S C U
P"
1190 PRINTSTRING$(64,140);
1200 PRINT"The New York Yacht Club could not defeat Australia II
neither on shore or off and had to surrender the cup to syndi
cate head Alan Bond and the Australian crew."
1210 PRINT"Australia II owed much of its success to the controve
rsial 'Winged Keel' as well as its sturdy management."
1220 PRINT"The Australian Yacht earned a good reputation as well
as a few nicknames - One of these being 'The wonder from Down
Under'."
1230 PRINT:PRINT"It is now some time after that glorious day and
you must defend the Australia's Cup. Australia II will not be c
ompeting as it is now in a museum, stuffed, along with other spor
ting greats such as Dennis Lillee & Robert de Castella."
1240 PRINTSTRING$(64,140);PRINT@979,"Press any key to continue"
;
1250 IFPEEK(15359)=0THEN1250
1260 CLS:PRINT" The A U S T R A L I A ' S C U
P":PRINTSTRING$(64,140);
1270 PRINT"The challengers for this years cup were from the Unit
ed States, England and New Zealand. The Australian trials have a
lready been held to decide the defender and the challengers have
all competed to decide the challenger."
1280 PRINT"The Australias Cup will be awarded to the yacht that
wins the best of seven races to be held in the forthcoming wee
k."
1290 PRINTSTRING$(64,140);
1300 PRINT"Up to 6 people can bet on the outcome of the seven ra
ces. The names of the Challenging and Defending yachts will be
displayed along with the odds for each."
1310 PRINT:PRINT"Australia's Cup (C) 10/83 by Carl Cranstone. Al
l Rights Reserved";
1320 PRINT@979,"Press any key to start.";
1330 IFPEEK(15359)=0THEN1330
1340 RETURN
1350 '*****

```

```

70 DATA 149, 149, 32, 149, 130, 131, 32, 149, 32, 149, 32, 149, 165, 160,
133, 32, 129, 32, 149, 32, 149, 149, 148, 149, 131, 131, 129, 1
65, 160, 133, 152, 137, 144, 129, 32, 129, 149, 32, 149, 130, 15
0, 32, 32, 129, 32, 131, 163
80 DATA 133, 152, 129, 32, 131, 131, 129, 32, 32, 32, 32, 32, 32
, 32, 32, 32, 134, 163, 132, 32, 133, 32, 129, 32, 32, 149,
32, 32, 133, 32, 32, 129, 32, 32, 32, 32, 32, 32, 32, 131, 3
2, 32, 32, 32, 32, 140, 32, 32
90 DATA 131, 32, 32, 32, 32, 32, 140, 32, 32, 139, 32, 32, 32, 3
2, 32, 32, 32, 139, 32, 136, 149, 32, 32, 149, 32, 130, 131,
32, 134, 131, 148, 152, 131, 32, 131, 131, 129, 134, 131, 148,
144, 131, 148, 130, 131, 32, 160
100 DATA 174, 32, 141, 174, 132, 32, 130, 32, 183, 179, 129, 144
, 32, 149, 130, 131, 32, 152, 131, 129, 151, 131, 148, 130, 131,
32, 131, 163, 133, 32, 149, 32, 32, 129, 32, 150, 131, 148, 150
, 131, 148, 130, 131, 32, 150
110 DATA 131, 148, 130, 163, 133, 131, 129, 32, 150, 163, 148, 1
57, 129, 149, 130, 131, 32, 184, 184, 144, 174, 174, 132, 128, 1
28, 128, 164, 181, 132, 155, 159, 145, 128, 129, 128, 156, 157,
132, 179, 183, 149, 128, 129, 32

```

\*\*\* 32K/DISK GRAFX \*\*\*

```

10 CLS:CLAR2000:READL:DIPL(L),T$(L)
20 FORX=1TOL:READL(X):NEXT:FORX=1TOL:FORX=1TOL(X):READC:T$(X)=T$
(X)+CHR$(C):NEXTV,X
30 FORX=1TOL:PRINT$(X-1)*64,T$(X):NEXT
40 DATA3,19,19,19
50 DATA150,131,132,32,151,131,148,32,152,137,144,32,151,131,129,
32,165,160,133,149,136,148,32,151,167,32,157,140,149,32,151,1
29,32,32,152,137,144,130,131,32,32,129,32,129,32
60 DATA129,32,129,32,129,32,32,32,129,32,129
70 PRINT@192,STRING$(64,131):PRINT@108,"Bob Wilson Software";
80 PRINT@256,CHR$(31)"One moment please:CMD'F",DELETE 10-70
90 CLEAR50:CLEARMEM-8500:DEFINTA-Z:DEFSTRW,P:GOTO210
100 A$=INKEY$:IFA$=" "THEN100ELSERETURN
110 CP=FIX(Y/3)*64+FIX(X/2)+15360:RETURN
120 SET(X,Y):FORZ=1T08:A=PEEK(14368):B=PEEK(14368):IFA=0ANDB=0TH
ENNEXTZ:RESET(X,Y):FORZ=1T08:A=PEEK(14400):B=PEEK(14368):IFA=0AN
DB=0THENNEXTZ:GOSUB190:IFA=0THEN120ELSERETURN
130 IFM=0THENSET(X,Y)ELSESET(X,Y)
140 GOSUB150:GOTO120
150 IFAANDBTHENY=Y+(Y>0)*1:RETURN
160 IFAAND16THENY=Y-(Y<44)*1:RETURN
170 IF(AAND32)OR(BAND16)THENX=X+(X>0)*1:RETURN
180 IF(AAND64)OR(BAND64)THENX=X-(X<127)*1:RETURN
190 A$=INKEY$:IFA$=" "THENA$=" "
200 A=INSTR("CDETX",A$):IFA$=" "THENRETURNELSEA=INSTR("cdetx",A$):RET
URN
210 W1=CHR$(31):W3="["
+CHR$(08)+CHR$(09)+",":W2=W1+"GRAFX MODE : % % : use ARRO
WS or < > ESC CTRL"
220 X$="ABCDEFGHIJKLMNPOQRSTUVWXYZ ?!.,:;1234567890##$%&'*~:abc
defghijklmnopqrstuvwxyz":DIMIT$(LEN(X$),3),SS$(3360),6$(25),P(25)
,U(15)

```

```

1860 DATAUnion Jack,Royal I,Britannia II,Victorious,Unsinkable,B
odyline '32,Victory '83,Britain I,Britain II,England III
1870 DATAKiwi I,Kiwi II,Auckland I,Wellington II,New Zealand I,I
ntrepid II,Restless Native,N.Z. Challenge,Keelhaul,Deadly Weapon
1880 OUT254,116:DS=259:CS=579:DD=1:CD=1:CLS:PRINT@0," Australi
a's Cup (c) 1983 by Carl Cranstone ** Demo Mode **":PRINTSTRING$
(64,131);
1890 DE$="

```

```

** Australia's Cup ** - ** Press <ENTER> to start **
- ** For up to six players **

```

```

1900 FORT=1T050:PRINT@ND(832)+128,".":NEXT
1910 X=RND(1):ONXGOSUB370,380
1920 IFPEEK(15359)<>0THENRUN
1930 ZZ=ZZ+1:IFZZ=LEN(DE$)THENZZ=0ELSEPRINT@960,MID$(DE$,ZZ,62)
:GOTO1910
1940 V=V+1:IFV=2THENRUNELSE1910

```

\*\*\* L2/16K AUSTRALIA'S CUP (AUSCUP/LNW) \*\*\*

```

10 MODE2:OUT254,116:PCLS:FLS
20 COLOR4:LINE0,0,159,11,SET,BF
30 COLOR2:LINE0,48,159,79,SET,BF
40 COLOR1:LINE0,80,159,88,SET,BF
50 COLOR6:LINE0,104,159,139,SET,BF
60 COLOR3:LINE0,140,159,147,SET,BF
70 COLOR7:LINE128,0,159,191,SET,BF
80 GOTO80
90 REM PSAVE"AUSCUP/GRF"

```

\*\*\* 32K/DISK GRAFX INIT PROGRAM \*\*\*

TRS-80/SYSTEM-80

```

10 CLS:CLAR3000:OPEN"O",1,"BIGLTRS"
20 FORX=1T0 414:READY:PRINT#1,Y;NEXT:CLOSE:END
30 DATA 152, 137, 144, 157, 140, 149, 129, 32, 129, 151, 131, 14
8, 151, 131, 148, 131, 131, 32, 150, 131, 132, 149, 32, 144, 130
, 131, 32, 151, 131, 148, 149, 32, 149, 131, 131, 32, 151, 131,
129, 151, 131, 32, 131, 131, 129
40 DATA 151, 131, 129, 151, 129, 32, 129, 32, 32, 150, 131, 132,
149, 136, 148, 130, 131, 32, 149, 32, 149, 151, 131, 149, 129,
32, 129, 130, 151, 32, 32, 149, 32, 130, 131, 32, 130, 131, 149,
144, 32, 149, 130, 131, 32, 149
50 DATA 152, 129, 151, 164, 32, 129, 32, 129, 149, 32, 32, 149,
32, 32, 131, 131, 129, 157, 152, 149, 149, 32, 149, 129, 32, 129
, 181, 32, 149, 149, 137, 149, 129, 32, 129, 150, 131, 148, 149,
32, 149, 130, 131, 32, 151, 131
60 DATA 148, 151, 131, 32, 129, 32, 32, 150, 131, 148, 149, 164,
149, 130, 131, 129, 151, 131, 148, 151, 167, 32, 129, 32, 129,
150, 131, 132, 146, 131, 148, 130, 131, 128, 131, 151, 129, 32,
149, 32, 32, 129, 32, 149, 32

```



```

650 PRINT@988,X-C:IF (PEEK(X)<>32ANDPEEK(X)<>128) THEN660ELSENEXT
X:PRINT@896,W1"Screen is empty: TRY AGAIN";:FORD=110500:NEXT:GO
10250
660 B=X-C:L=INT(B/64):R=B-L*64:IFR=0THENR=64ELSEL=L+1
670 G=5:P(1)=-1 CLS:CLREAR2000:READL:DIML(L),T$(L)
680 P(2)=-2 FORX=1TOL:READL(X):NEXT:FORX=1TOL:FORY=1TOL(X):READC
:T$(X):T$(X)+CHR$(C):NEXTY,X
690 P(3)=-3 FORX=1TOL:PRINT@X-1)*64,T$(X):NEXTX":P(4)=-4 DATA
+STR$(L):P(5)=-5 DATA
700 FORX=1TOL:SB=C+(X-1)*64:G$(X)=-:POKEVARPTR(G$(X))+2,INT(SB/
256):POKEVARPTR(G$(X))+1,SB-INT(SB/256)*256:POKEVARPTR(G$(X)),64
:NEXT
710 PRINT@960,W1"Writing program";:FORX=1TOL:PRINT@976,W1X:FOR
Y=64T0250STEP-1:PRINT@980,Y:V$=MID$(G$(X),Y,1):IFV$=CHR$(32) THENN
EXTY
720 POKEVARPTR(G$(X)),Y:P(4)=P(4)+", "+STR$(Y):FORZ=1TOY:TL=TL+1:
PRINT@984,USING"###";TL:P(G)=P(G)+STR$(ASC(MID$(G$(X),Z,1)))+"",
":IFLEN(P(G))>220THENGOSUB760:G=6+1:P(G)=STR$(G)+ " DATA"
730 NEXTZ,X:GOSUB760:K=G:G=3:GOSUB760:S$="S":SN$=7:GOSUB790
740 CLS:PRINT@256,"PROGRAM COMPLETE:
Program is : "K"lines Numbered 1 TO"K"
Graphics data : "TL"elements
Filespec : "F$
750 PRINT"
DUMPING PROGRAM :":OPEN"O",1,F$:FORX=1TOK:PRINT#1,P(X):NEXT:CLOSE
E:PRINT@960,"Hit any key";:GOSUB100:SN$=7:S$="R":GOSUB790:GOTO25
0
760 P(G)=LEFT$(P(G),LEN(P(G))-1):RETURN
770 PRINT@960,W1"Save as screen # <1-6>";:GOSUB100:SN$=INSTR("12
3456",A$):IFSN$THENS$="S":GOTO790ELSE770
780 PRINT@960,W1"Recall screen # <1-6>";:GOSUB100:SN$=INSTR("123
456",A$):IFSN$THENS$="R"ELSE780
790 DEFUSR=VARPTR(U(8)):IF$S="S" THENU(9)=15360:U(11)=VARPTR(SS$(
(SN%-1)*480))ELSEU(9)=VARPTR(SS$(SN%-1)*480)):U(11)=15360
800 J=USR(0):RETURN
810 DATA8448,15552,4352,15360,256,768,-20243,201
820 DATA8448,0,4352,0,256,960,-20243,201
830 CLS:CMD"DIR 0":END

```

\*\*\* L2/16K LVAR UTILITY \*\*\*

TRS-80/SYSTEM-80

##LVAR=NAME##

BY TIM FISH

9 CAVENDISH RD

COLLIERS WOOD

LONDON SW19 2ET

ENGLAND

```

00001 ;
00002 ;
00003 ;
00004 ;
00005 ;
00006 ;
00007 ;
00008 ;

```

```

230 OPEN"1",1,"BIGLRS":PRINT"Initializing HUGE letters":FORX=1T
OLEN(X$):FORY=1T03:FORZ=1T03:INPUT#1,C:T$(X,Y)=T$(X,Y)+CHR$(C):N
EXTZ:NEXTY,X:CLOSE
240 FORX=0T015:READU(X):NEXT:J=0:CLS:S$="S":FORSNK=1T07:GOSUB790
:NEXT
250 PRINT@960,"<C>lear <G>ratx <H>uge <S>ave <R>ecall <P>ro
gram <E>nd";:GOSUB100
260 S=INSTR("CGHPSRE",A$):IFSTHENONGOSUB300,310,470,620,770,780,830
,830
270 S=INSTR("cgbpsre",A$):ONGOSUB300,310,470,620,770,780,830
280 GOTO250
290 ONAGOTO320,360,370,380,250
300 CLS:RETURN
310 X=0:Y=0
320 PRINT@960,USINGW2;"CURSOR";
330 GOSUB110:CH=PEEK(CP)
340 POKECP,32:SET(X,Y):FORZ=1T08:A=PEEK(14400):B=PEEK(14368):IFA
=0ANDB=0THENNEXTZ:POKECP,CH:FORZ=1T08:A=PEEK(14400):B=PEEK(14368
):IFA=0ANDB=0THENNEXTZ:GOSUB190:IFA=0THEN340ELSE290
350 POKECP,CH:GOSUB150:GOTO330
360 PRINT@960,USINGW2;"DRAW";:M=0:GOSUB120:SET(X,Y):GOTO290
370 PRINT@960,USINGW2;"ERASE";:M=1:GOSUB120:GOTO290
380 PRINT@960,W1"TEXT Mode : <ENTER> to Exit";:GOSUB110
390 CH=PEEK(CP):POKECP,140:FORZ=1T010:A$=INKEY$:IFA$=""THENNEXT:
POKECP,CH:FORZ=1T010:A$=INKEY$:IFA$=""THENNEXT:GOTO390
400 POKECP,CH:ONINSTR(W3,A$)GOTO430,440,450,460,450,460
410 IFA$=CHR$(13) THEN420ELSEPOKECP,ASC(A$):CP=CP+1:GOTO390
420 XY=CP-15360:Y=FIX(XY/64)*3+1:X=2*(XY-INT(XY/64))*64:GOTO320
430 CP=CP+(CP\15423)*64:GOTO390
440 CP=CP-(CP\16256)*64:GOTO390
450 CP=CP+(CP\15360)*1:GOTO390
460 CP=CP-(CP\16319)*1:GOTO390
470 L=0:GOTO500
480 GOSUB600:PRINT@+64,"XXX";:GOSUB100:S=INSTR(X$,A$):IFSTHEN55
0
490 S=INSTR(X$,A$):IFSTHEN550
500 S=INSTR(CHR$(13)+CHR$(08)+CHR$(24)+CHR$(91),A$):IFS=0THEN480
ELSEGOSUB590:ONGOTO510,520,590,530
510 L=INT((L+16)/16)*16-1:GOTO560
520 L=L+(L>0)*1:GOTO540
530 L=L+(L>15)*16
540 S=27:GOSUB600:GOSUB610:GOTO480
550 GOSUB610
560 L=L+1:IFL<80THEN480
570 DEFUSR=VARPTR(U(8)):J=USR(0):PRINT@760,W1L=L-16
580 PRINT@960,W1"SHIFT/BACKSPACE to Exit";:GOTO480
590 PRINT@+64,"":RETURN
600 Q=INT(L/16)*192+(L-INT(L/16))*16)*4:RETURN
610 FORY=1T03:PRINT@+ (Y-1)*64,T$(S,Y):NEXT:RETURN
620 TL=0:C=15360:PRINT@960,W1"Enter 1 letter to identify program
";
630 A$=INKEY$:IFA$="" THEN630ELSE$=ASC(A$):IFS<65ORS>90THEN630
640 F$="TITLE"+A$+"/GFX":PRINT@960,W1"LOCATING END OF DATA";:FOR
X=16319T0CSTEP-1

```

```

00009 ;
00100 LD 7E41H
00102 LD A,0C3H ;C3=JP OP
00104 LD (418EH),A ;ADDRESS OF "NAME" DOS EXI
T
00106 LD HL,START
00108 LD (418FH),HL
00110 JP 1A33H
00120 LD BC,(40F9H) ;START OF VLT
00122 LD A,(COUNT)
00124 CP 0
00126 JR NZ,NOT15
00128 CALL 2BH
00130 OR A
00132 JR Z,AGN
00134 LD A,15
00136 DEC A
00138 LD (COUNT),A
00140 PUSH BC
00142 POP HL
00180 LD DE,(40FBH) ;END OF VLT+1
00200 SBC HL,DE ;TEST FOR END
00220 JR Z,OUT
00240 INC BC
00260 INC BC
00280 LD A,0DH ;LINE FEED
00300 CALL 33H
00320 LD A,(BC)
00340 CALL 33H
00360 DEC BC
00380 LD A,(BC)
00400 CALL 33H
00420 DEC BC ;POINT TO CODE LENGTH
00440 LD A,(BC)
00460 LD (40AFH),A
00480 CP 2 ;INTEGER?
00500 JR NZ,NOTINT
00520 LD A,%
00540 CALL 33H
00560 LD A,%
00580 CALL 33H
00600 INC BC
00620 INC BC
00640 INC BC
00660 LD A,(BC) ;VALUE MSB
00680 LD (4121H),A ;WRA1
00700 INC BC
00720 LD A,(BC)
00740 LD (4122H),A
00760 LD HL,(4121H)
00780 BIT 7,H ;TEST FOR NEG VALUE
00800 JR Z,POS
00820 LD DE,0
00840 EX DE,HL
00860 XOR A ;CLEAR CARRY

```

---

```

00880 SBC
00900 LD
00920 CALL 33H
00940 PUSH
00960 CALL 0FAFH
00980 BC
01000 BC
01020 JR NEXT
01040 OUT
01060 NOTINT
01080 JR NZ,NOTSNG
01100 LD A,%
01120 CALL 33H
01140 INC BC
01160 INC BC
01180 INC BC ;POINT TO VALUE
01200 LD D,4
01220 LD HL,4121H
01240 LD A,(BC)
01260 LD (HL),A
01280 INC HL
01300 INC BC
01320 DEC D
01340 JR NZ,LOOP1
01360 LD A,0 ;NO PRINT USING
01380 BC
01400 CALL 0FBEH ;CONVERT TO ASCII & STORE
01420 CALL 28A7H ;PRINT IT
01440 POP BC
01460 JR STEP
01480 HALFVY
01500 NOTSNG
01520 JR NZ,STRING
01540 LD A,%
01560 LD 33H
01580 CALL 33H
01600 INC BC
01620 INC BC
01640 INC BC
01660 LD D,8
01680 LD HL,411DH
01700 LD LOOP1
01720 CP 3
01740 JR NZ,OUT
01760 LD A,%
01780 CALL 33H
01800 LD A,%
01820 CALL 33H
01840 LD A,%
01860 CALL 33H
01880 INC BC
01900 INC BC
01920 INC BC
01940 LD A,(BC)
01960 PUSH AF

```

```

01980 INC BC
02000 LD A, (BC) ;MSB STRING ADDRESS
02020 LD L,A
02040 INC BC
02060 LD A, (BC)
02080 LD H,A
02100 POP AF
02120 CP 0
02140 JR Z,SKIP ;IF NULL STRING
02160 PUSH BC
02180 LD B,A
02200 LD A, (HL)
02220 CALL 33H
02240 INC HL
02260 DJNZ LOOP2
02280 POP BC
02300 INC BC
02320 LD A, ' '
02340 CALL 33H
02360 JR HALFVY
02380 DEF B 15
02400 END INIT

START END ENTRY
7E41 7F2C 7E41

7E41: 3E C3 32 8E 41 21 4F 7E 22 8F 41 C3 33 1A ED 4B
7E51: F9 40 3A 2C 7F FE 00 20 08 CD 2B 00 B7 28 FA 3E
7E61: 0F 3D 32 2C 7F C5 E1 ED 5B FB 40 ED 52 28 4A 03
7E71: 03 3E 0D CD 33 00 0A CD 33 00 0B 0A CD 33 00 0B
7E81: 0A 32 AF 40 FE 02 20 32 3E 25 CD 33 00 3E 3D CD
7E91: 33 00 03 03 03 0A 32 21 41 03 0A 32 22 41 2A 21
7EA1: 41 CB 7C 28 0C 11 00 00 EB AF ED 52 3E 2D CD 33
7EB1: 00 C5 CD AF 0F C1 03 18 99 76 FE 04 20 20 3E 3D
7EC1: CD 33 00 03 03 03 16 04 21 21 41 0A 77 23 03 15
7ED1: 20 F9 3E 00 C5 CD BE 0F CD A7 28 C1 18 D9 FE 08
7EE1: 20 14 3E 23 CD 33 00 3E 3D CD 33 00 03 03 16
7EF1: 08 21 1D 41 18 D5 FE 03 20 BF 3E 24 CD 33 00 3E
7F01: 3D CD 33 00 3E 22 CD 33 00 03 03 0A F5 03 0A
7F11: 6F 03 0A 67 F1 FE 00 28 0A C5 47 7E CD 33 00 23
7F21: 10 F9 C1 03 3E 22 CD 33 00 18 B1 0F

20 POK&H40B1, &H00:POK&H40B2, &H80:POK&H407F, PEEK (&H40A0)
30 CLS:CL&A50:DEFINT A-Z:CP=&H4023:CC=PEEK(CP):POKECP, 32
40 PRINT@72, "EDTASM ASSEMBLER SOURCE UTILITY FOR MODEL III"
50 PRINT TAB(19); "NEWDOS80 Version 2.0"
60 PRINTTAB(15); "( Copyright 1982 T. Domigan )":GOSUB470
70 DEFUSR0=&H404E:DEFUSR1=&H4070:BU=&H8000:MD=&H4056
80 POK&H4024, &H01:POK&H4210, &H28:POK&H4214, &H04
90 H=&CHR$(244)+CHR$(245)+CHR$(246)+CHR$(32)
100 CLS:PRINT:PRINTTAB(29); "MENU"
110 PRINT:PRINT" 1. DISK FILE TO MEMORY BUFFER
2. CASSETTE FILE TO MEMORY BUFFER
3. MEMORY BUFFER TO CASSETTE FILE
4. MEMORY BUFFER TO DISK FILE"
120 PRINT" 5. CLEAR MEMORY BUFFER
6. EXIT PROGRAM AND REPAIR BASIC"
130 GOSUB460
140 IF (EQ<1)OR(EQ>6) THENGOTO130ELSEIF(EQ<6) THEN170
150 POK&H4214, &H00:POKECP, CC:POK&H40B1, &HFF:POK&H40B2, &HFF
160 POK&H40A0, PEEK (&H407F):POK&H40A1, &HFF:CLS:CLOSE:CLS:END
170 ONEGOTO180, 240, 300, 350, 430
180 ME=BU:CLS:PRINT@398, "DISK FILE TO MEMORY ROUTINE"
190 PRINT@582, "Enter Filespec ( with Extension ) ==> ";
200 LINEINPUTFS:OPEN"R", 1, FS$, 1:PRINT@720, "OPENING FILE ";H$;FS
$
210 FIELD1, IASAS
220 FORIX=1TOLOF(1):GETI, IX:Y=ASC(IAS$):ME=ME+1:POKEME, Y:NEXTIX
230 CLOSE:PRINT@855, "END OF FILE":GOTO290
240 CLS:PRINT@398, "CASSETTE FILE TO MEMORY ROUTINE"
250 PRINT@590, "Press ENTER when cassette is ready":GOSUB460
260 POK&H150:POK&H2, 205:POK&H3, 53:POK&H4, 2:POK&H5, 119
270 X=USR0(P)
280 PRINT@845, "Cassette file has been READ to memory"
290 FORT=1TO1000:NEXTT:GOTO100
300 CLS:PRINT@398, "MEMORY TO CASSETTE FILE ROUTINE"
310 POK&H135:POK&H2, 126:POK&H3, 205:POK&H4, 100:POK&H5, 2
320 PRINT@590, "Press ENTER when cassette is ready":GOSUB460
330 X=USR0(P)
340 PRINT@847, "Cassette file has been WRITTEN":GOTO290
350 CLS:PRINT@398, "MEMORY TO DISK FILE ROUTINE"
360 PRINT@585, "Enter Filespec ( with Extension ) ==> ";
370 LINEINPUTFS:OPEN"O", 1, FS$:SM=BU:SO=BU+1
380 PRINT@722, "OPENING FILE ";H$;FS$
390 S1=PEEK(SM):S2=PEEK(SO):PRINT@1, CHR$(S2);
400 IF (S2<>26) THENGOTO400ELSEIF (S1=13) THENCLOSE
410 PRINT@855, "END OF FILE":GOTO290
420 SM=SM+1:SO=SO+1:GOTO390
430 CLS:PRINT@400, "CLEAR BUFFER ROUTINE"
440 Y=USR1(Q)
450 PRINT@721, "BUFFER IS NOW CLEAN":GOTO290
460 EQ$=INKEY$:IFEQ$=" " THEN460ELSEEQ$=VAL (EQ$):RETURN
470 FORDT=&H404ETO&H407E:READD:POK&H1, D:NEXTDT:RETURN
480 DATA205, 66, 48, 33, 1, 128, 243, 205, 0, 2, 0, 0, 35, 254
490 DATA 26, 32, 247, 43, 43, 126, 254, 13, 40, 4, 35, 35, 24, 236
500 DATA205, 248, 1, 201, 33, 1, 128, 17, 2, 128, 1, 0, 112
510 DATA62, 0, 119, 237, 176, 201

```

\*\*\* MODEL 3 SOURCE UTILITY \*\*\*

TRS-80

SOURCE/BAS

10 REM  
EDTASM SOURCE-TAPE UTILITY FOR MODEL III NEWDOS80 V2.0  
(copyright 1982 T. Domigan)

# NEXT MONTH'S ISSUE

Next month's issue will contain at least the following programs plus the usual features and articles. An (80) after a program title indicates that the program will be for TRS-80 Model 1/3 or System 80/Video Genie. A (CC) indicates that the program will be for the TRS-80 Colour Computer and (HP) that the program is for the Hitachi Peach.

**CRICKET (CC)**

Join in the Summer fun and play your own World Series Cricket on your COCO. Complete with batsmen, bowler and fielders. A game for two players.

**ALIEN CHASE (HP)**

Another "get them before they get you" game for Peach users. You pursue the Aliens around the screen and are in turn pursued by them.

**AUTOMATIC DIRECTORY (80) — 32K DISK**

Some of the newer DOS's have a facility to speed up file manipulation from the Directory. AUTOMATIC DIRECTORY gives you this facility from earlier DOS's. You may KILL, LOAD or LIST a file simply by placing the cursor against that file name on the DIRectory display and pressing the appropriate key. In addition, you may assign keys to particular files so you may load such a file with a single keystroke.

**DISK DIRECTORY RECORDER (MODEL 3)**

About the time you start on your second box of diskettes, you run into the problem of keeping track of all those files. It seems a shame to use pen and paper to do this when you have a perfectly good computer there. Disk Directory recorder stores a sorted catalogue of all files showing the name of each file, its extensions and the name of the disk on which it may be found. It is thus a simple matter to update your catalogue as you add and delete new files.

**FILM COSTING (80) L2/16K**

Whilst we do not expect too many of our readers are involved in processing large quantities of photographic film, this program which calculates processing costs does illustrate some interesting programming points and could probably be adopted to a variety of similar uses.

# CASSETTE/DISK EDITION INDEX

The cassette edition of MICRO-80 contains all the applicable software listed each month, on cassette. For machine language programs copies of both the source and object file are provided. All programs are recorded twice. Level 1 programs can only be loaded into a Level 2 machine if the 'Level 1 in Level 2' program from the MICRO-80 Software Library — Vol. 1 is loaded first.

**Note:** System 80/Video Genie computers have had different tape-counters fitted at different times. The approximate start positions shown are correct for the very early System 80 without the volume control or level meter. They are probably incorrect for later machines. The rates for a cassette subscription are printed on the inside front cover of each issue of the magazine.

The disk edition contains all applicable programs which can be executed from disk. Level 1 disk programs are saved in NEWDOS format. Users require the Level 1/CMD utility supplied with NEWDOS+ or NEWDOS 80 version 1.0 to run them.

Side 1	Type	I.D.	Disk Filespec	Approx. Start Position		
				CTR-41	CTR-80	System 80
GRAFX INIT PROG	32K DISK	I	INIT/BAS	18	10	6
GRAFX INIT PROG	32K DISK	I	INIT BAS	47	26	11

APPLICATION FOR PUBLICATION  
OF A PROGRAM  
IN MICRO-80

Date .....

To MICRO-80  
SOFTWARE DEPT.,  
P.O. BOX 213,  
GOODWOOD, S.A. 5034

Please consider the enclosed program for  
publication in MICRO-80.

Name .....

Address .....

Postcode .....

\*\*\* CHECK LIST \*\*\*

Please ensure that the cassette or disk is clearly marked with your name and address, program name(s), Memory size, Level I, II, System 1 or 2, Edtasm, System, etc. The use of REM statements with your name and address is suggested, in case the program becomes separated from the accompanying literature.

Ensure that you supply adequate instructions, notes on what the program does and how it does it, etc.

For system tapes, the start, end, and entry points, etc.

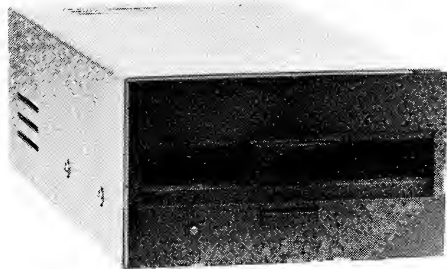
The changes or improvements that you think may improve it.

Please package securely — padabags are suggested — and enclose stamps or postage if you want your cassette or disk returned.

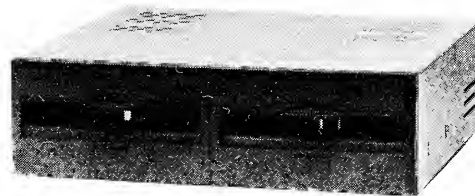




# SAVE A PACKET ON MICRO-80's DISK DRIVE PACKAGES FOR TRS-80 MODEL 1 AND SYSTEM 80 MICROCOMPUTERS



**SINGLE DRIVE PACKAGE from ... \$499**



**DUAL DRIVE PACKAGE from ... \$874**

Bigger volume means lower cost price, which we are passing on to you. Avoid the annoying bundle of cables, wires and separate boxes. MICRO-80 is now offering our well-proven MPI disk drives in attractive, self-contained single or dual-drive cabinets complete with internal power supply. Our drive Ø and dual-drive packages also include the appropriate version of DOSPLUS and dual-drive cable.

*The best news of all is the specially reduced package prices ...*

**SAVE \$23 — \$107 over our already low prices!**

*Choose the appropriate system from the table below:*

DRIVE TYPE	No. of Tracks	No. of Heads	Capacity	Dosplus Version	Price	* Saving
<b>DRIVE Ø</b>						
1 x MPI B51	40	1	100K	3.4	\$499	\$137.95
1 x MPI B52	40	2	200K	3.4	\$639	\$97.95
1 x MPI B92	80	2	400K	3.4	\$799	\$107.95
<b>DRIVE 1</b>						
1 x MPI B51	40	1	100K	—	\$415	\$33.00
1 x MPI B52	40	2	200K	—	\$525	\$23.00
1 x MPI B92	80	2	400K	—	\$695	\$23.00

\*Represents the saving compared with buying all the items included in the package separately

•Drive Ø package includes one bare disk drive, self-contained single-drive cabinet/power supply as illustrated, two drive cable and the version of DOSPLUS indicated.

•Drive 1 package includes one bare disk drive and self-contained single-drive cabinet/power supply as illustrated.

*If it's a dual-drive system you need, then take advantage of our dual-drive package and  
**SAVE a further \$40 on the price of two single-drive packages ...***

DRIVE TYPE	No. of Tracks	No. of Heads	Capacity	Dosplus Version	Price
2 x MPI B51	40 ea	1 ea	2 x 100K	3.4	\$874
2 x MPI B52	40 ea	2 ea	2 x 200K	3.4	\$1125
2 x MPI B92	80 ea	2 ea	2 x 400K	3.4	\$1454

Dual-drive package includes two bare disk drives, self-contained dual-drive cabinet/power supply as illustrated, two drive cables and the version of Dosplus indicated.

NOTE: All 40 track drives are completely compatible with 35 track operating systems such as TRSDOS. DOSPLUS allows you to realise an additional 14% capacity compared with TRSDOS. Under DOSPLUS 3.4, 80 track drives can read 35/40 track diskettes.

All disk drive components are still available separately:

**BARE DRIVES** — MPI drives offer the fastest track-to-track access time (5 milliseconds) available. All drives are capable of operating in double density for 80% greater storage capacity.

	Price	Freight			
MPI B51 40 track, single-head, 100K	\$349	\$5.00	Self-contained, single drive cabinet/power supply	\$99	\$5.00
MPI B52 40 track, dual-head, 200K	\$449	\$5.00	Self-contained, dual-drive cabinet/power supply	\$135	\$5.00
MPI B92 80 track, dual-head, 400K	\$619	\$5.00	Two drive cable	\$39	\$2.00
Separate, dual-drive power supply	\$85		Four drive cable	\$49	\$2.00

Prices are FOB Adelaide. Add \$5.00 freight for single drive package, \$10.00 for dual-drive package. Prices are in Australian dollars. Freight is road freight anywhere in Australia.

All items carry a 90-day parts and labour warranty. Repairs to be carried out in our Adelaide workshops.



# MICRO-80

---

## LEVEL 2 ROM ASSEMBLY LANGUAGE TOOLKIT by Edwin Paay FOR TRS-80 MODEL 1, MODEL 3 AND SYSTEM 80/VIDEO GENIE

This is a new package consisting of two invaluable components:

- **A ROM REFERENCE** Manual which catalogues, describes and cross-references the useful and usable ROM routines which you can incorporate into your own machine language or BASIC programs.
- **DEBUG**, a machine language disassembling debugging program to speed up the development of your own machine language programs. DEBUG is distributed on a cassette and may be used from disk or cassette.

Part 1 of the ROM REFERENCE manual gives detailed explanations of the processes used for arithmetical calculations, logical operations, data movements etc. It also describes the various formats used for BASIC, System and Editor/Assembly tapes. There is a special section devoted to those additional routines in the TRS-80 Model 3 ROM. This is the first time this information has been made available, anywhere. Differences between the System 80/Video Genie are also described. Part 1 is organised into subject specific tables so that you can quickly locate all the routines to carry out a given function and then choose the one which meets your requirements.

Part 2 gives detailed information about each of the routines in the order in which they appear in the ROM. It describes their functions, explains how to use them in your own machine language programs and notes the effect of each on the various Z80 registers.

Part 2 also details the contents of system RAM and shows you how to intercept BASIC routines. With this knowledge, you can add your own commands to BASIC, for instance, or position BASIC programs in high memory — the only restriction is your own imagination!

The Appendices contain sample programmes which show you how you can use the ROM routines to speed up your machine language programs and reduce the amount of code you need to write.

DEBUG: Eddy Paay was not satisfied with any of the commercially available debugging programs, so he developed his own. DEBUG: allows you to single-step through your program; has a disassembler which disassembles the next instruction before executing it or allows you to bypass execution and pass on through the program, disassembling as you go; displays/edits memory in Hex or ASCII; allows Register editing; has the ability to read and write System tapes and all this on the bottom 3 lines of your screen, thus freeing the rest of the screen for program displays. Four versions of DEBUG are included in the package to cope with different memory sizes.

**The best news of all is the price. The complete Level 2 ROM ASSEMBLY LANGUAGE TOOLKIT is only:**

- Aus. \$29.95 + \$2.00 p&p
- UK £18.00 + £1.00 p&p

**SPECIAL OFFER TO OWNERS OF THE LEVEL II ROM REFERENCE MANUAL ...**

**UPGRADE TO THIS ASSEMBLY LANGUAGE TOOLKIT FOR ONLY \$19.95!**

**Send back your original Level II ROM Reference Manual plus a cheque, money order or Bankcard authorisation for \$19.95 plus \$2.00 p&p and we will send you the new ASSEMBLY LANGUAGE TOOLKIT**

---

# MICRO-80